Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Gašperin

Podpora procesiranju kompleksnih dogodkov v okviru standardne platforme IoT

magistrsko delo

študijski program druge stopnje
računalništvo in informatika

Mentor: prof. dr. Marko Bajec

Ljubljana, 2016

Zahvaljujem se mentorju, prof. dr. Marku Bajcu in as. dr. Slavku Žitniku za vse nasvete in pomoč pri izdelavi magistrske naloge. Prav tako se želim zahvaliti Simonki Gregurovič, za odpravo slovničnih napak.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 IOT platforme **3**

2.1 Komercialne platforme 4

2.2 Odprtokodne platforme 9

3 Orodja za procesiranje kompleksnih dogodkov **15**

3.1 Microsoft StreamInsight 16

3.2 ATSD - Axibase Time Series Database 16

3.3 WSO2 CEP 17

3.4 SQLstream Blaze 17

3.5 Esper 18

4 Rešitev za procesiranje kompleksnih dogodkov v IoT platformi OM2M **21**

4.1 Uporabljene knjižnice 21

4.2 Tehnične in vsebinske zahteve 26

4.3 Delovanje nadgradnje 28

4.4 Vzpostavitev platforme OM2M z nadgradnjo za procesiranje kompleksnih dogodkov 30

5 Praktičen prikaz delovanja **57**

5.1 Simulator senzorja pritiska in pospeškometra 58

5.2 Pravilo CEP 59

5.3 Aplikacija Monitor 60

5.4 Aplikacija za platformo Android 61

5.5 Struktura sporočil 61

5.6 Evalvacija sistema 68

6 Splošnost razvitega pristopa **81**

6.1 Repozitorij vtičnika in vključitev v uradni repozitorij platforme OM2M 83

7 Zaključek **85**

Slike

2.1 Primerjalna matrika komercialnih in odprtokodnih platform IOT. 4
2.2 Arhitektura platforme ThingWorx. 5
2.3 Platforma Jasper. 6
2.4 Shema platforme Xively. 7
2.5 Pregled platforme OpenMTC. 8
2.6 Arhitektura platforme AllJoyn. 10
2.7 Komponente platforme WSO2. 11
2.8 Platforma IoTivity. 12
2.9 Glavne komponente platforme OM2M. 13

3.1 Obdelava toka podatkov in korelacija v realnem času. 18

4.1 Povezave med uporabljenimi knjižnicami. 22
4.2 Primerjava Jetty z ostalimi spletnimi strežniki [25]. 23
4.3 Primerjava H2 Database Engine z ostalimi podatkovnimi bazami [27]. 25
4.4 Delovanje nadgradnje za procesiranje kompleksnih dogodkov. . 29
4.5 Inštalacija novih aplikacij. 30
4.6 Dodajanje Tycho repozitorija. 31
4.7 Inštalacija aplikacije Tycho. 32
4.8 Kloniranje OM2M repozitorija git v orodje Eclipse. 33
4.9 Urejanje povezave z repozitorijem git platforme OM2M. 34
4.10 Master veja repozitorija OM2M. 35

4.11 Vključitev projekta po uspešnem kloniranju. 36
4.12 Statusi vseh naloženih vtičnikov. 37
4.13 Vpis v spletno mesto platforme OM2M. 38
4.14 Drevesna struktura produkta IN-CSE. 39
4.15 Kreiranje novega vtičnika v orodju Eclipse. 40
4.16 Kreiranje novega vtičnika v orodju Eclipse (drugi del). 40
4.17 Predloge kreiranega vtičnika. 41
4.18 Drevesna struktura kreiranega vtičnika. 42
4.19 Ureditev datoteke "build.properties". 42
4.20 Menjava tipa projekta v Maven projekt. 43
4.21 Izbira nadrejenega vtičnika. 44
4.22 Rezultat izgradnje platforme OM2M. 45
4.23 Stanje kreiranega vtičnika v zagnani platformi OM2M. 46
4.24 Dodajanje knjižnice za procesiranje kompleksnih dogodkov v orodje Eclipse. 48
4.25 Dodajanje knjižnice za procesiranje kompleksnih dogodkov v projekt. 49
4.26 Implementacija razreda DataInterface. 50
4.27 Metode, ki implementirajo razred DataInterface. 51
4.28 Modifikacija razreda Monitor. 52
4.29 Zagon platforme OM2M. 53
4.30 Imenik platforme OM2M in datoteka start.bat. 54
4.31 Zagon kreiranega vtičnika v platformi OM2M. 54
4.32 Kreiranje pravila CEP (GUI). 55
4.33 Kreiranje pravila CEP (programsko). 55
4.34 Podatkovni modeli kreiranega vtičnika platforme OM2M. . . . 56

5.1 Shema praktičnega prikaza delovanja. 58
5.2 Simulator senzorja. 59
5.3 Aplikacija Monitor. 60
5.4 Aplikacija za platformo Android. 61
5.5 Struktura sporočil znotraj platforme OM2M. 62

5.6 Evalvacija platforme OM2M brez knjižnice CEP 69
5.7 Povprečna zakasnitev prejema podatka v aplikacijo, ki je naročena na podatkovni model glede na časovni interval pošiljanja podatkov na platformo OM2M pri enem, dvajsetimi in tridesetimi povezanimi senzorji 71
5.8 Evalvacija platforme OM2M s knjižnico CEP 72
5.9 Povprečna zakasnitev prejema podatka v aplikacijo Monitor ter CEP Monitor glede na časovni interval pošiljanja podatkov na platformo OM2M pri enem uporabljenem CEP pravilu za 1 in 50 povezanimi senzorji 74
5.10 Povprečna zakasnitev prejema podatka v aplikacijo Monitor ter CEP Monitor glede na časovni interval pošiljanja podatkov na platformo OM2M pri dvajsetih uporabljenih CEP pravilih za 1 in 50 povezanih senzorjev 77
5.11 Povprečna zakasnitev prejema podatka v aplikacijo Monitor ter CEP Monitor glede na časovni interval pošiljanja podatkov na platformo OM2M pri petdesetih uporabljenih CEP pravilih za 1 in 50 povezanih senzorjev 80

6.1 Podatkovna pot med vtičnikom platforme OM2M in nadgradnjo CEP 82
6.2 Podatkovna pot med vtičnikom platforme OM2M in nadgradnjo CEP v primeru bolj splošnega pristopa implementacije . 83

7.1 Beleženje zgodovine v podatkovni bazi in prikaz podatkov . . 86

Tabele

5.1 Evalvacija sistema brez knjižnice CEP 70
5.2 Evalvacija sistema s knjižnico CEP z enim definiranim pravilom CEP 73
5.3 Evalvacija sistema s knjižnico CEP z dvajsetimi definiranimi pravili CEP 75
5.4 Evalvacija sistema s knjižnico CEP s petdesetimi definiranimi pravili CEP 78

Seznam uporabljenih kratic

kratica	opis
IOT	Internet of things
AE	Application Entity
NSE	Network Services Entity
IN	Infrastructure node
ASN	Application service node
MN	Middle node
CSE	Common Services Entity
CEP	Complex event processing
DSL	Domain Specific Language
EPL	Event Processing Language
SQL	Structured Query Language
ATSD	Axibase Time Series Database
ANSI	American National Standards Institute
OM2M	Open Source platform for M2M communication
EPL	Eclipse Public License
MPL	Mozilla Public License
BLOB	Binary Large Object
SOAP	Simple Object Access Protocol
GPL	General Public License
RCP	Rich Client Platform
RAP	Remote Application Platform
IPE	Interworking Proxy Unit

kratica	**opis**
oBIX	Open Building Information Xchange
XML	Extensible Markup Language
GCM	Google Cloud Messaging
GUI	Graphical User Interface
LWM2M	Lightweight M2M
3GPP	3rd Generation Partnership Project

Povzetek

Naslov: Podpora procesiranju kompleksnih dogodkov v okviru standardne platforme IoT

Internet stvari predstavlja enega izmed najpomembnejših in hitro razvijajočih se trendov današnjega časa na področju računalništva in informatike. Sistemska programska oprema, ki omogoča procesiranje podatkov in povezovanje naprav v internetu stvari, je platforma IoT. Teh je izjemno veliko, le redke pa so odprtokodne in temeljijo na odprtih standardih kot je oneM2M. Ena takih platform je platforma OM2M, ki pa ne podpira funkcionalnosti procesiranja kompleksnih dogodkov in je pomembna za številne scenarije IOT. V ta namen smo se odločili, da bomo platformo OM2M nadgradili s to funkcionalnostjo, in sicer tako, da bo celotna arhitektura še vedno skladna s standardom. Razširitev ne posega v obstoječo platformo, je ne spreminja in uporablja izključno odprtokodne knjižnice. Razvijalcem omogoča veliko svobodo pri uporabi in je zasnovan tako, da podpira veliko možnih scenarijev, ki se uporabljajo v svetu interneta stvari. Nadgradnja je na voljo v obliki vtičnika ali knjižnice, ki se ob uporabi preprosto vključi v projekt platforme OM2M.

Ključne besede: Internet stvari, procesiranje kompleksnih dogodkov, OM2M - Odprtokodna platforma za M2M komunikacije, Espertech.

Abstract

Title: Support of complex event processing in standard IoT platform

Internet of things is one of the most important and rapidly evolving trends of our time in computer science and informatics. Software, which enables data processing and device connectivity in the internet of things is called IoT platform. There are plenty of them in the market, but only few of them are open source and based on open standards as oneM2M. One of those platforms is called OM2M, but unfortunately it does not support complex event processing, which is very important for a number of scenarios in IoT. For this purpose, we decided that we will upgrade OM2M platform with that functionality in such a way that the entire architecture will still be consistent with open standards. Implemented extension does not change the OM2M platform in any way and it uses exclusively open source libraries. It allows developers to use it in different ways and is designed to support many scenarios used in the world of internet of things. The upgrade is available as a plug-in or library, which can be simply included in the project of OM2M platform.

Keywords: IOT - Internet of things, CEP - Complex Event Processing, OM2M - Open Source platform for M2M communication, Espertech.

Poglavje 1

Uvod

Na enega izmed najpomembnejših in hitro razvijajočih se trendov današnjega časa na področju računalništva in informatike se uvršča internet stvari (ang. Internet of Things, IoT)[29]. Če za internet velja, da med seboj povezuje računalnike, za IoT velja, da med seboj povezuje poljubne "stvari iz fizičnega sveta". Hiter razvoj na področju komunikacijskih protokolov ter tehnologij, kot so RFID in NFC, namreč omogoča, da se že skoraj slehrni objekt lahko priključi na internet, prejema, oddaja in tudi procesira podatke. Analitske hiše kot je Garner, napovedujejo zelo pomembno vlogo interneta stvari v razvoju informacijske družbe. Vlaganja v ta trend naj bi do leta 2020 po nekaterih ocenah dosegla tudi 2 trilijona dolarjev [2]. Dodaten pospešek k razvoju IoT prispeva tudi velik napredek na področju razvoja senzorjev, ki so postali cenovno izjemno ugodni in energetsko varčni [3].

Platform interneta stvari je izjemno veliko [4, 5], le redke pa so odprto kodne in temelječe na odprtih standardih, kot so oneM2M, OIC - Open Interconnect Consortium, ThreadGroup, AllSeen Alliance [6]. Ena takih je platforma OM2M [7, 8], ki sledi standardu oneM2M. Pri našem preučevanju smo ugotovili, da ta platforma ne podpira funkcionalnosti procesiranja kompleksnih dogodkov (ang. Complex Event Processing, CEP) [9, 10], ki je pomembna za številne IoT scenarije. V okviru magistrske naloge smo platformo OM2M razširili tako, da podpira tudi to funkcionalnost.

Prvi del magistrske naloge opisuje komercialne platforme interneta stvari ThingWorx [11], Jasper [12], Xively [13], Axiros [14], OpenMTC [15], odprtokodne platforme IoT AllJoyn [16], WSO2 [17], IoTivity [18], OM2M [7] ter orodja in tehnologije, ki se danes uporabljajo za procesiranje kompleksnih dogodkov Microsoft StreamInsight [39], Axibase Time Series Database [20], WSO2 CEP [21], SQLstream Blaze [22] in Esper [23].

V nadaljevanju bomo podrobneje preučili spletni strežnik Jetty [24], relacijsko podatkovno bazo H2 Database Engine [26] ter orodje za procesiranje kompleksnih dogodkov Esper [23]. Opisali bomo razvito nadgradnjo platforme OM2M, knjižnice, ki so bile uporabljene v nadgradnji, tehnične in vsebinske zahteve, ki smo si jih zadali pred samim razvojem ter opis in navodila za vključitev nadgradnje v platformo OM2M. Sledi praktičen prikaz delovanja nadgradnje s povezanim simulatorjem senzorja pritiska in pospeškometra, kjer bomo s pomočjo CEP pravil lovili kompleksne dogodke, jih pošiljali v aplikacijo Monitor in preko storitve GCM [28] pošiljali na pametni telefon uporabnika. Prikazali bomo tudi strukturo sporočil, ki se pošiljajo preko posameznih enot sistema.

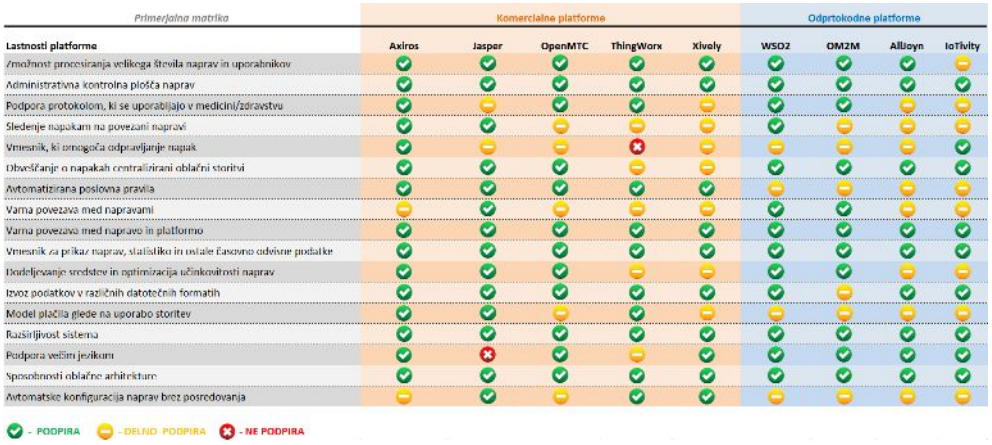
Nazadnje bomo naredili še evalvacijo platforme OM2M z in brez vključene nadgradnje. Zanimalo nas bo kako hitro platforma procesira podatke, jih posreduje naprej na naročeno napravo ter kako se ta hitrost spreminja s povečevanjem števila postavljenih CEP pravil, števila povezanih naprav na platformo in časovnega intervala pošiljanja podatkov na platformo.

Poglavje 2

IOT platforme

IOT oziroma Internet of things [29, 30] je omrežje fizičnih naprav, vozil, stavb in ostalih objektov, ki vsebujejo elektronska vezja, senzorje in možnost povezave z omrežjem. IoT tem objektom omogoča, da so zaznana in kontrolirana skozi obstoječa infrastrukturna omrežja. Vsak objekt je dostopen in točno določen v njegovem infrastrukturnem omrežju. En objekt z drugim lahko komunicira brez človekovega posredovanja ter na podlagi izmenjanih informacij proži prej definirane akcije. Strokovnjaki ocenjujejo, da bo do leta 2020 v IoT povezanih več kot 50 bilijonov naprav.

V zadnjem času so pričakovanja in potenciali na področju IoT precejšnja, zato je bilo v tem obdobju razvitih več sto platform, ki omogočajo komunikacijo z različnimi napravami, shranjevanje, pregled in obdelavo njihovih podatkov. Večinoma so to komercialne rešitve, nekaj izmed teh pa je tudi odprtokodnih. V tem poglavju bomo opisali prednosti in slabosti ter medsebojno primerjali nekatere izmed najbolj razširjenih platform IoT [31] (Slika 2.1).



Slika 2.1: Primerjalna matrika komercialnih in odprtokodnih platform IOT.

2.1 Komercialne platforme

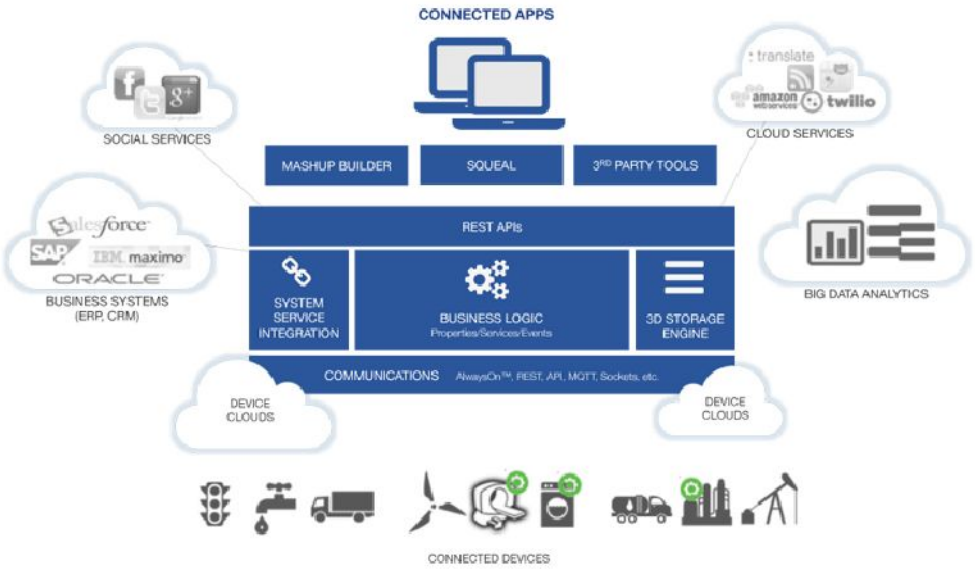
Skoraj največ razvitih platform na področju IoT so komercialne rešitve. Večina izmed njih ponujajo administrativni vmesnik za kontrolo naprav, zmožnost procesiranja velikega števila naprav in uporabnikov, avtomatizacijo poslovnih pravil, uporabniški vmesnik za prikaz statistike in ostalih časovno odvisnih podatkov, razširljivost sistema ter oblačno arhitekturo [32, 33].

2.1.1 ThingWorx

ThingWorx [11] je široko sprejeta platforma, ki se ponaša kot prva platforma IoT, ustvarjena za hitro gradnjo in zagon aplikacij povezanega sveta. Poenostavlja ustvarjanje aplikacij za povezane naprave in s tem daje razvijalcem orodja, ki jih potrebujejo za povezovanje, ustvarjanje ter analiziranje njihovih naprav.

Platforma ima zmožnost procesiranja velikega števila podatkov, podpira protokole, ki se uporabljajo v zdravstvu in medicini, omogoča možnost avtomatiziranja poslovnih pravil, avtomatsko rekonfiguriranje naprav glede na določene pogoje ter plačevanje storitev glede na njihovo uporabo.

Kljub temu, da platforma ponuja široko paleto storitev, ima nekaj pomanjkljivosti. Razvijalcem je delo na tej platformi nekoliko oteženo, saj ne vsebuje veliko orodij za razhroščevanje ter le delno podpira varno komunikacijo med napravami.



Slika 2.2: Arhitektura platforme ThingWorx.

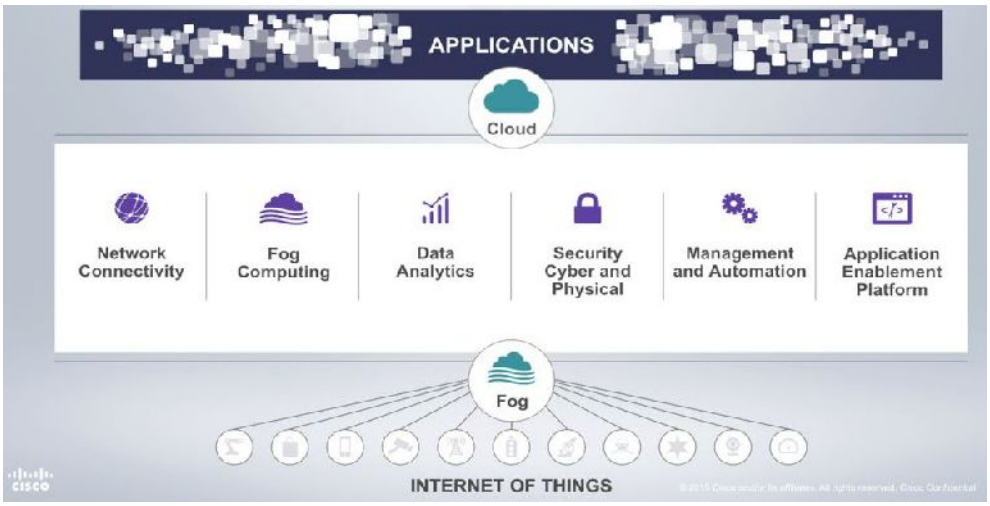
2.1.2 Jasper

Jasper [12] je platforma, ki omogoča optimiziacijo in avtomatizacijo faz v življenjski dobi IoT storitve. S tem omogoča optimalno izkoriščanje povezanih naprav in aplikacij, pospeši čas uveljave uporabnikove storitve na trgu, omogoča kontrolo nad napravami v realnem času ter zagotavlja visoko zanesljivost storitev.

Platforma Jasper zagotavlja zmožnost velikega procesiranja podatkov, omogoča obveščanje o napakah centralizirani storitvi v oblaku, sledenjem napak na povezanih napravah, podpira varno komunikacijo med napravami ter omogoča dodelitev virov željenim napravam. Tako kot platforma ThingWorx omogoča možnost avtomatiziranja poslovnih pravil, avtomatsko re-

konfiguriranje naprav glede na določene pogoje ter plačevanje storitev glede na njihovo uporabo.

Glavna slabost platforme je, da ne podpira veliko jezikov, slabo podpira razhroščevanje težav ter protokole, ki se uporabljajo v medicini.



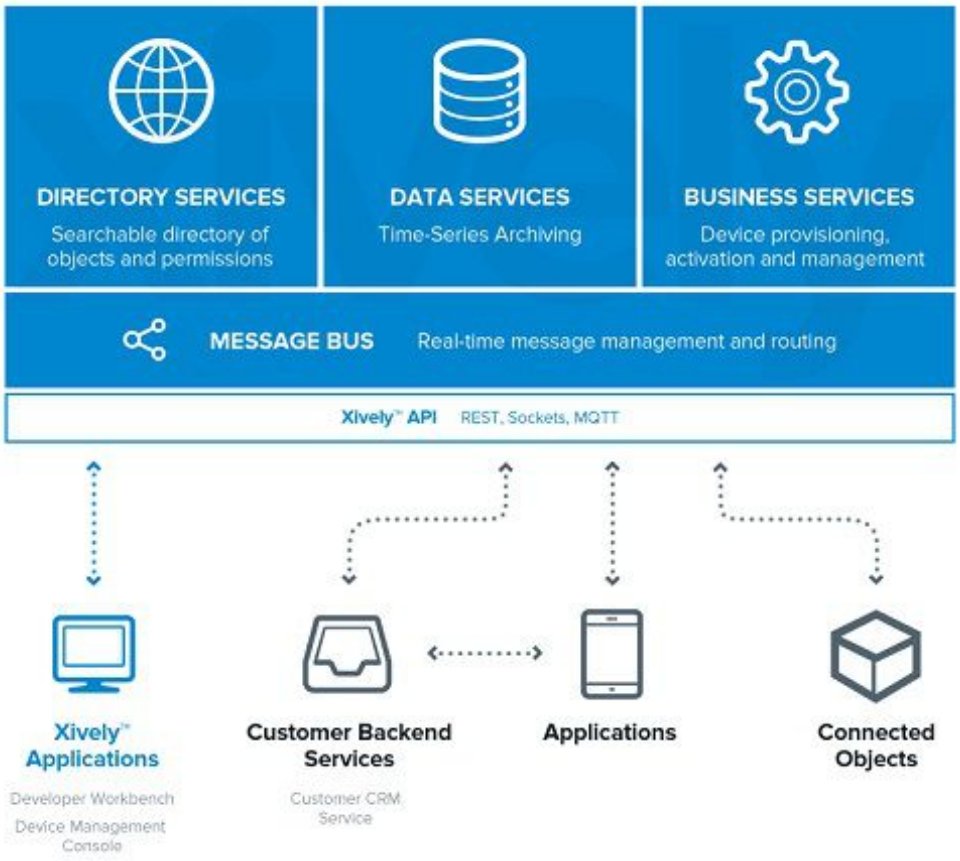
Slika 2.3: Platforma Jasper.

2.1.3 Xively

Xively [13] formalno znan tudi kot Cosm ali Pachube je globalno podjetje, ki ponuja programsko opremo kot storitev. Sloni na LogMeInovi oblačni platformi, imenovani Gravity, ki premore procesiranja več kot 255 miljonov naprav in uporabnikov. Ponujajo platformo IoT in aplikativno rešitev podjetjem, ki se ukvarjajo s povezovanjem naprav in storitev.

Xively omogoča povezovanje velikega števila naprav, avtomatizacijo poslovnih pravil, varno povezovanje naprav na platformo, možnost izvažanja različnih poročil, podpora več jezikom ter avtomatsko rekonfiguriranje naprav glede na določene pogoje.

Slabosti platforme so slabo podpiranje orodij za sledenje napakam in razhroščevanje, le delna podpora medicinskim protokolom, varne komunikacije med napravami ter nezmožnost optimizacije dodeljevanja virov.



Slika 2.4: Shema platforme Xively.

2.1.4 Axiros

Axiros [14] ponuja široko paleto visoko priznanih rešitev za upravljanje s storitvami in napravami. Omogoča spremljanje in interakcijo s katerokoli povezano napravo v skoraj realnem času. Ponaša se z hitro orkestracijo povezanih naprav z minimalno količino vloženega dela. Platforma je primerna za telekomunikacijska podjetja ter prav tako za katerokoli storitveno podjetje.

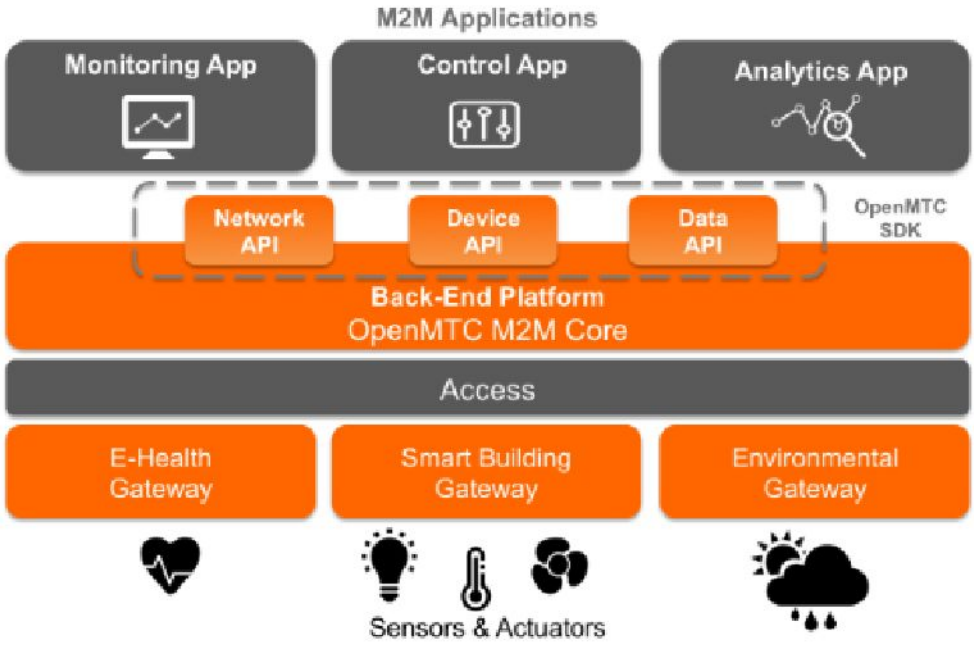
Platforma ponuja orodja za razhroščevanje in omogoča sledenje napakam, ki se zgodijo na sistemu, podpira protokole, ki se uporabljajo v medicini, omogoča avtomatizacijo poslovnih pravil, varno komunikacijo naprav z platformo, optimizacijo dodeljevanja virov, plačevanje storitev glede na njihovo

uporabo ter omogoča podporo več jezikov.

Platforma le delno omogoča varno komunikacijo med napravami ter avtomatsko prepoznavanje povezanih naprav in konfiguracijo.

2.1.5 OpenMTC

OpenMTC [15] je platforma, ki ponuja oblačno rešitev za izvajanje različnih raziskav in razvijanje IoT aplikacij. Na trgu je že od aprila 2012 in ponuja povezovanje različnih naprav iz različnih domen (transport, logistika, e-zdravje, ustanove itd.), združevanje prejetih informacij, posredovanje, monitoriranje ter analizo. Platforma je združjiva z različnimi mednarodnimi standardi (ETSI M2M, oneM2M, LWM2M in 3GPP).



Slika 2.5: Pregled platforme OpenMTC.

2.2 Odprtokodne platforme

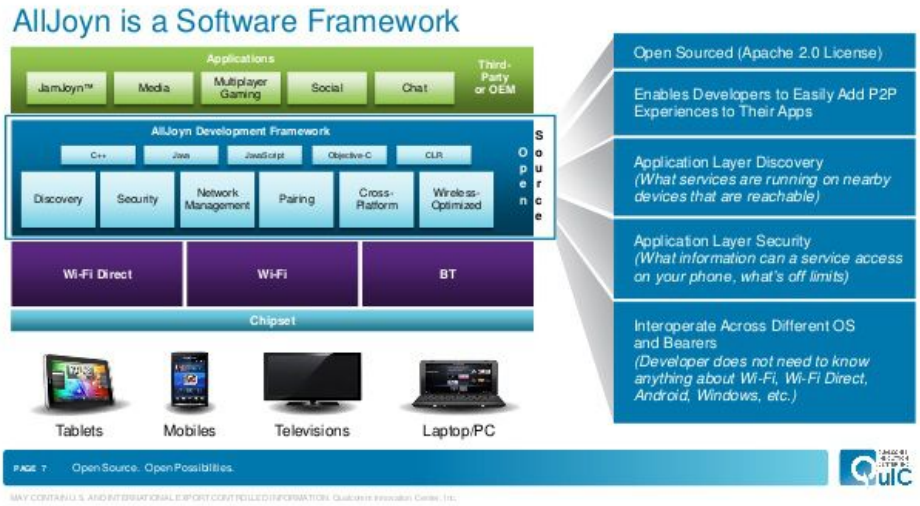
Odprtokodnih IoT platform je na trgu zelo malo. Še manj pa je takih, ki bi obenem temeljile na široko sprejetih standardih IoT. Večina izmed njih ponuja spletni uporabniški vmesnik z administrativno ploščo za urejanje povezanih naprav, varno povezovanje in komunikacijo naprav s platformo, grafične predstavitve časovno odvisnih podatkov, zajetih iz različnih naprav, paleto datotečnih formatov za izvoz podatkov ter sposobnost oblačne arhitekture.

2.2.1 AllJoyn

AllJoyn [16] je odprtokodna platforma, ki temelji na standardu AllseenAliance [34]. Napravam in aplikacijam omogoča medsebojno odkrivanje in komunikacijo. Je neodvisna od operacijskega sistema ter omogoča razvijalcem pisanje aplikacij neodvisno od transportnega sloja, proizvajalca in brez dostopa do interneta.

Platforma ima podprto obveščanje o napakah centralizirani storitvi v oblaku, podpira možnost velikega procesiranja podatkov, zagotavlja varno povezavo med napravo in platformo, vsebuje administrativno ploščo za pregled, vizualizacijo in druge funkcije nad časovno odvisnimi podatki. Omogoča izvoz informacij v različnih datotečnih formatih, podporo osnovnih storitev v oblakih ter podporo več jezikom.

Slabost platforme je, da le delno podpira sledenje napak, ki se zgodijo na povezanih napravah, varni komunikaciji med napravami, dodeljevanju virov izbranih naprav ter avtomatske rekonfiguracije naprav glede na predhodno določene pogoje.



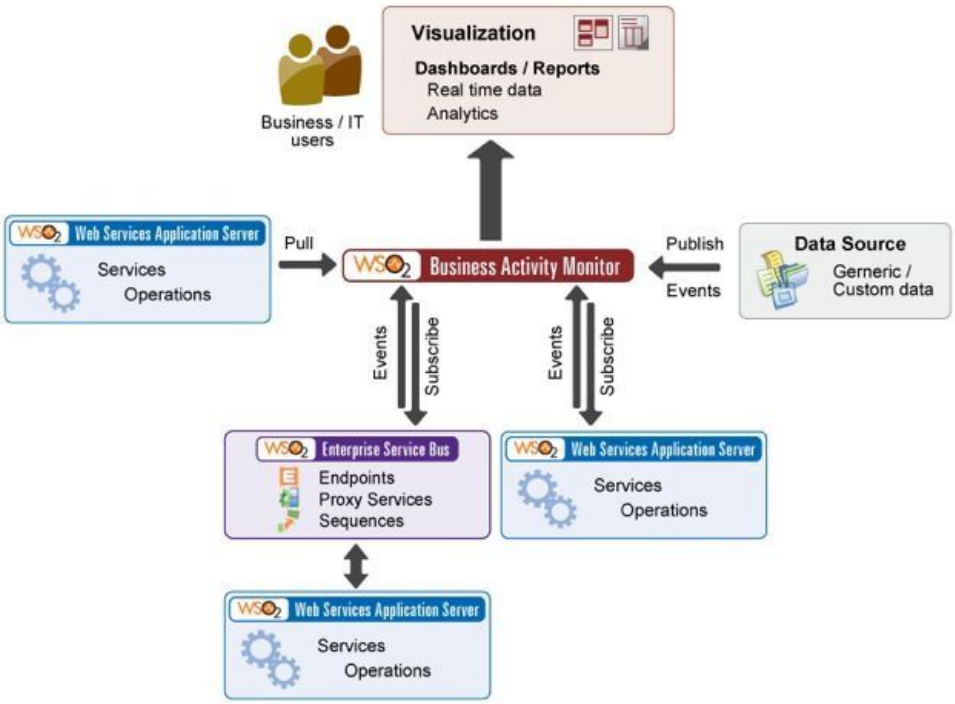
Slika 2.6: Arhitektura platforme AllJoyn.

2.2.2 WSO2

WSO2 IoT strežnik [17] ponuja dokončno, varno, odprtokodno rešitev za upravljanje z povezanimi napravami. Ponuja strežniško stran arhitekture interneta stvari za proizvajalce naprav. Platformo je mogoče prilagoditi in nadgraditi s WSO2 podatkovno analitičnim strežnikom. Uporabnikom bo v celoti na voljo do konca leta 2016.

WSO2 strežnik poleg osnovnih storitev IoT podpira protokole, ki se uporabljajo v medicini, sledenje napak na povezanih napravah in obveščanje le teh oblačni storitvi, varno povezavo med vsemi napravami v omrežju, dodeljevanje virov izbranim napravam ter avtomatsko rekonfiguracijo naprav.

Strežnik ne podpira v celoti vmesnika za iskanje in razhroščevanje napak na sistemu, avtomatizacije poslovnih pravil ter avtomatske konfiguracije povezanih naprav.



Slika 2.7: Komponente platforme WSO2.

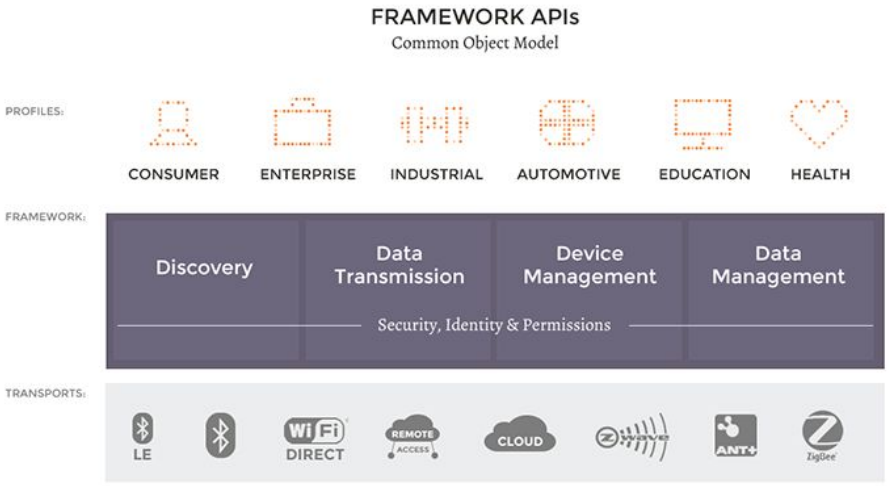
2.2.3 IoTivity

IoTivity [18] je odprtokodna platforma, ki omogoča celovito povezljivost med napravami. Projekt je bil narejen z namenom združitve odprtokodne skupnosti za hitrejši razvoj platforme in storitev, potrebnih za povezljivost milijone naprav. Projekt IoTivity je implementacija standardnih specifikacij OIC (Open Interconnect Consortium) [35] in je deljen pod licenco Apache 2.0.

Projekt IoTivity ima dobro podprta orodja za razhroščevanje in obveščanje napak na napravah, omogoča varno povezovanje naprav s platformo, omogoča analizo in izvoz podatkov v različnih formatih, razširljivost sistema ter podporo več jezikom.

Sistem le delno podpira avtomatizacijo poslovnih pravil, protokole, ki se uporabljajo v medicini, varno povezljivost med napravami, optimizacijo

dodeljevanja virov sistema in avtomatsko konfiguracijo povezanih naprav ter urejanje.



Slika 2.8: Platforma IoTivity.

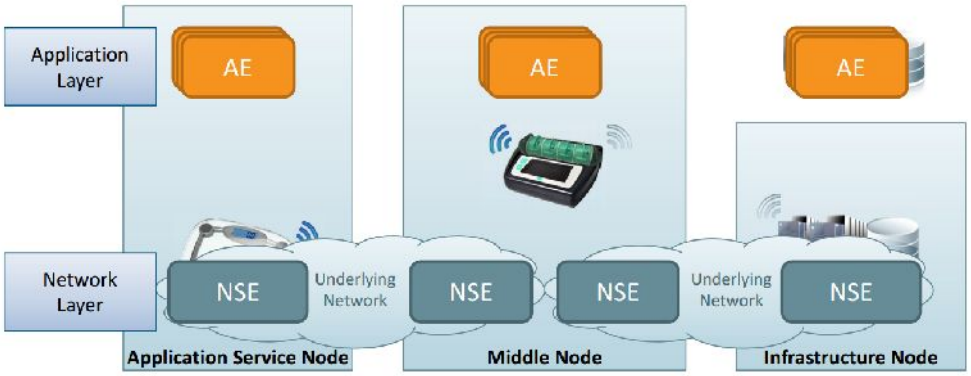
2.2.4 OM2M

Platforma OM2M [7, 8], znana tudi kot Open Source platform for M2M communication, je odprtokodna implementacija široko sprejetega IoT standarda OneM2M [36] in SmartM2M [37], razvita s strani LAAS-CNRS (Laboratory for Analysis and Architecture of Systems). Zagotavlja horizontalno povezljivo platformo, katere dele se lahko namesti na M2M strežnik (Infrastructure node), prehod (Middle node) ali napravo (Application service node). Vsak del platforme zagotavlja varno povezljivost, proženje obvestil, trajnost, medsebojno skladnost, upravljanje z napravami ter druge storitve, ki so v skladu s OneM2M standardom.

Platforma temelji na REST storitvah, ki zagotavljajo primitivne postopke za avtentikacijo naprav, odkrivanje virov, registracijo aplikacij, upravljanje

naprav, sinhrono in asinhrono komunikacijo, avtorizacijo in organizacijo skupin naprav. Razvita je v programskem jeziku Java, ki teče na OSGi Equinox okolju, zaradi česar je platforma zelo razširljiva preko lastno razvitih vtičnikov. Zgrajena je kot Eclipsov projekt, ki uporablja Maven in Tycho. Vsak vtičnik ponuja specifično funkcionalnost in ga je mogoče daljinsko zagnati, ustaviti, posodobiti in odstraniti brez potrebe po ponovnem zagonu. Platforma je sestavljena iz treh glavnih komponent (Slika 2.9):

- Infrastructure node (IN-CSE): Deluje kot strežnik, ki vsebuje en CSE (Common Services Entity) in opcijsko AE (Application Entity)
- Middle node (MN-CSE): Deluje kot prehod, ki vsebuje vsaj en CSE (Common Services Entity) in opcijsko AE (Application Entity)
- Application service node (ASN-CSE): Deluje kot naprava, ki vsebuje en CSE (Common Services Entity) in vsaj en AE (Application Entity)



Slika 2.9: Glavne komponente platforme OM2M.

Osnovna funkcionalnost platforme je posredovanje podatkov iz ene naprave na drugo. Deluje kot neke vrste usmerjevalnik, s tem da se posredovani podatki shranjujejo v podatkovno bazo H2, ki je implementirana v hitrem spominu. Pregled povezanih naprav in podatkov ter sam pregled platforme

omogoča preko spletnega vmesnika, ki je implementiran s pomočjo knjižnice Jetty.

Sporočila, ki se prenašajo preko spletnih storitev, so v formatu XML zgrajena po standardu oBIX (Open Building Information Xchange) [38]. To je standard, ki se uporablja za generiranje XML sporočila v sistemih, kjer prevladuje komunikacija med napravami. Cilj standarda je poenotenje izmenjanih sporočil v spletnih storitvah na čim bolj preprost in varen način v vseh IoT sistemih.

Podrobna preučitev te platforme se nam zdi najbolj smiselna, saj je implementacija OneM2M standarda, ki je najbolj razširjen od štirih najpomembnejših standardov na področju IoT. Tudi nadgradnja te platforme je smiselna, saj trenutno ne podpira procesiranja kompleksnih dogodkov, je odprtokodna in je lahko razširljiva preko dodatnega vtičnika, brez poseganja v originalno implementacijo.

Poglavje 3

Orodja za procesiranje kompleksnih dogodkov

CEP oziroma Complex event processing [9, 10] je metoda kompleksnega procesiranja, sledenja in analiziranja toka informacij o stavareh, ki so se zgodile. Cilj procesiranja kompleksnih dogodkov je identifikacija takih dogodkov, ki so za nas pomembni (priložnosti, grožnje, nevarnosti) in odziv na le te v čim krajšem možnem času.

Vzemimo na primer napravo za merjenje krvnega pritiska, ki jo nosimo na telesu in jo priključimo na platformo IoT. Namesto stalnega spremljanja vrednosti krvnega tlaka bi želeli, da nas platforma opozori, v kolikor krvni tlak preseže neko mejno vrednost in na njej ostane dlje časa, medtem ko mi v resnici mirujemo .

Večina odprtokodnih platform IoT takšne funkcionalnosti sama po sebi ne podpira, je pa na trgu kar nekaj samostojnih programskih rešitev za procesiranje kompleksnih dogodkov. V tem poglavju bomo opisali nekatere izmed teh rešitev.

3.1 Microsoft StreamInsight

Tipične relacijske podatkovne baze temeljijo na poizvedbah, za potrebe IoT pa potrebujemo podatkovne baze, ki temeljijo na dogodkih. StreamInsight [39] je rešitev Microsofta za podporo procesiranju kompleksnih dogodkov. Je platforma za razvoj aplikacij, ki temeljijo na tehnologiji .NET in potrebujejo implementacijo procesiranja kompleksnih dogodkov.

StreamInsight je na voljo kot CEP strežnik, ki je sestavljen iz jedra in različnih adapterjev. Adapterji omogočajo razvijalcem kreirati vmesnike, preko katerih strežnik prejema podatke iz različnih virov in na katere proži akcije oziroma pošilja filtrirane informacije. Vsak dogodek, ki je prejet preko vmesnika, se zapiše v podatkovno bazo, hkrati pa se primerja z definiranimi pravili na strežniku za proženje akcij. V primeru, da se zaporedje dogodkov ujame v katero izmed postavljenih pravil, se akcija lahko sproži na določenem izhodnem vmesniku.

3.2 ATSD - Axibase Time Series Database

Axibase Time Series Database (ATSD) [20] je oblačna nerelacijska podatkovna baza, namenjena shrambi informacij, ki prihajajo iz neke infrastrukture. Oblikovana je posebej za shrambo in analiziranje velikih količin podatkov, shranjenih z visoko frekvenco zbiranja. Uporabnikom omogoča izluščevanje vrednosti iz podatkov, ki že obstajajo v njihovih infrastrukturah, spremljanje, identifikacijo in obveščanje o abnormalnih obnašanjih v produkcijskih sistemih, pregled podatkov preko administrativne plošče, izvoza statističnih podatkov ter služi kot centraliziran repozitorij za zbiranje podatkov.

Podatkovna baza privzeto vsebuje orodja za analitično postavljanje pravil, napovedovanje ter vizualizacijo. Orodje za napovedovanje vsebuje različne časovno odvisne algoritme, s katerimi omogoča napovedovanje možnih nevarnosti že v zgodnji fazi, orodje za vizualizacijo pa omogoča prikaz podatkov z različnimi grafi. Orodje za analitično postavljanje pravil je orodje, s katerim ATSD omogoča procesiranje kompleksnih dogodkov. Omogoča definira-

nje pravil, njihovega tipa, določanje časovnega okvirja, opis ter akcijo, ki se proži, ko se dogodek ujame v določeno pravilo.

3.3 WSO2 CEP

WSO2 Complex Event Processor [21] je odprtokodna platforma, ki omogoča identifikacijo najbolj pomembnih dogodkov in vzorcev iz različnih podatkovnih virov, analizira njihov vpliv in omogoča odzivanje na njih v realnem času.

V osrčju platforme je odprtokodni projekt Siddhi (Complex event processing engine) [40]. Konfiguracija je možna preko enostavnega, kompaktnega jezika, podobnega jeziku SQL, ki je namenjen kompleksnim poizvedbam z možnostjo poizvedovanja znotraj časovnega okvirja ter vzorca. Poizvedbe je možno spreminjati v času delovanja s pomočjo različnih predlog. Samo delovanje platforme ne zaseda veliko virov, prav tako ne prostora in je zmožno procesiranja zelo velikega števila podatkov.

WSO2 CEP platforma je na voljo kot samostojna aplikacija. Vsebuje spletni vmesnik za urejanje, pregled ter statistično predstavitev podatkov. Ima vgrajeno funkcionalnost za simuliranje dogodkov, podporo za časovno zahtevne poizvedbe, integracije IoT, rahroščevanje ter podporo za ostale WSO2 produkte, kar pomeni, da je platforma zelo razširljiva.

3.4 SQLstream Blaze

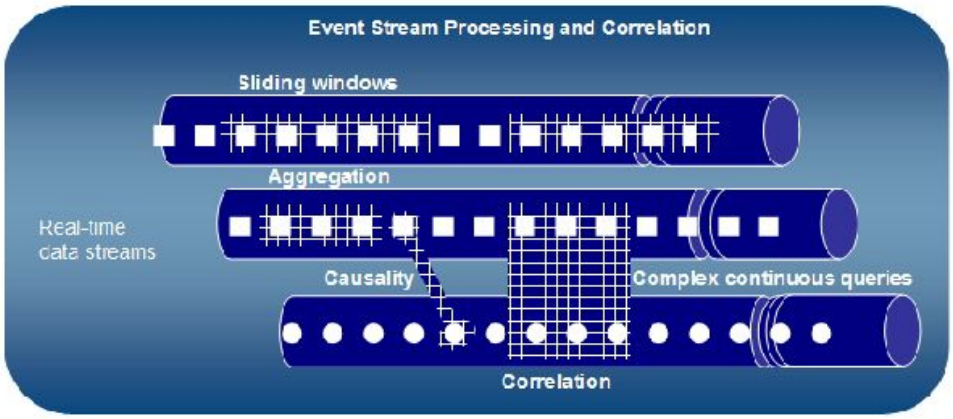
SQLstream [22] ponuja platformo za procesiranje podatkovnega toka, imenovano SQLstream Blaze. Omogoča analiziranje velikih količin podatkov, senzorjev in naprav v realnem času. Podatkovni tok je analiziran s pomočjo standardiziranega SQL jezika, standarda ANSI (American National Standards Institute) ter definiranega fiksnega ali premikajočega časovnega okna. Poizvedbe, ki so definirane na ta način, se ne izvršijo in zaključijo, ko so definirane, vendar se izvršijo vsakič, ko se pojavijo novi podatki za shranjevanje.

Platforma SQLstream Blaze je sestavljena iz orodij s-Server, s-Studio,

s-Dashboard, s-Visualizer in StreamLab, ki skupaj omogočajo vse omenjene funkcionalnosti.

3.5 Esper

Esper [23] je odprtokodna knjižnica, ki podpira procesiranje kompleksnih dogodkov (CEP) za aplikacije, izdelane v programskem jeziku Java in C#. Izdana je pod licenco GNU General Public License, verzije 2.0 (GPL-2.0) [41]. Omogoča hitrejše razvijanje aplikacij, ki procesirajo velike količine sporočil ali podatkov, analiziranje toka sporočil na različne načine ter odzivanje na njih glede na predefinirane pogoje. Ponuja DSL jezik (Domain Specific Language) za procesiranje dogodkov ter EPL jezik (Event Processing Language) za deklaracijo filtrov, združevanje podatkov čez več premikajočih se oken iz več prejetih podatkov (Slika 3.1).



Slika 3.1: Obdelava toka podatkov in korelacija v realnem času.

Primeri definicij EPL pravil:

- Pravilo vrne povprečno vrednost za vsak simbol iz zadnjih 100 zapisov:

```
select symbol, avg(price) as averagePrice
from StockTickEvent.win:length(100)
group by symbol
```

- Pravilo opozori na vsako delnico IBM s ceno, večjo od 80 v naslednjih 60 sekundah:

```
every StockTickEvent(symbol="IBM", price>80)
where timer:within(60 seconds)
```

- Pravilo vrne povprečno vrednost delnice v zadnjih 30 sekundah:

```
select avg(price)
from StockTickEvent.win:time(30 sec)
```

Implementacija knjižnice Esper v IoT platformo OM2M je najbolj smiselna, saj je izdelana tako, da jo je mogoče vgraditi v lastno aplikacijo, ponuja knjižnico za programski jezik Java, v katerem je izdelana tudi platforma OM2M, zaseže najmanj virov ter omogoča definiranje kompleksnih dogodkov s pomočjo deklarativnega jezika EPL.

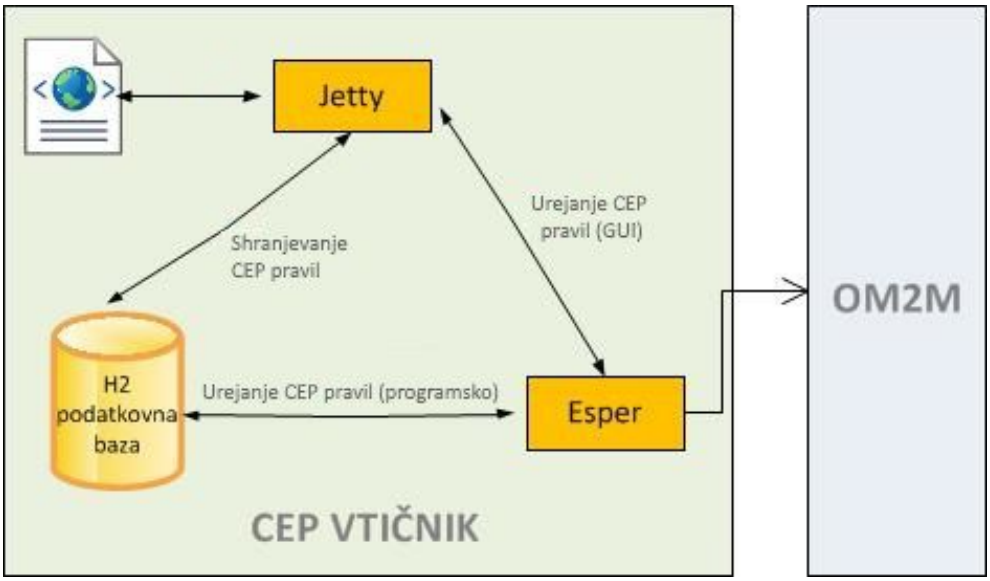
Poglavje 4

Rešitev za procesiranje kompleksnih dogodkov v IoT platformi OM2M

Pri podrobnem pregledu platforme OM2M smo ugotovili, da je platforma striktna implementacija OneM2M standarda in da tako kot večino odprtokodnih platform ne podpira procesiranja kompleksnih dogodkov. Ker ima standard OneM2M na trgu velike potenciale in ker je možnost procesiranja kompleksnih dogodkov pomembna za številne scenarije IoT, smo se odločili, da bomo v sklopu magistrske naloge platformo nadgradili s to funkcionalnostjo.

4.1 Uporabljene knjižnice

Rešitev za procesiranje kompleksnih dogodkov uporablja tri knjižnice Jetty, H2 Database Engine in Esper, ki so medsebojno povezane in skrbijo za procesiranje dogodkov, spletno prezentacijo nadgradnje in shranjevanje v podatkovno bazo (Slika 4.1).



Slika 4.1: Povezave med uporabljenimi knjižnicami.

4.1.1 Esper

Večino orodij za procesiranje kompleksnih pogojev je zasnovanih kot strežnik ali kot spletna storitev, knjižnica Esper pa je ena redkih orodij, ki omogoča procesiranje kompleksnih dogodkov kar znotraj aplikacije, v katero je knjižnica implementirana. Za našo nadgradnjo to pomeni transparentnost knjižnice, neodvisnost od ostalih sistemov in bolj kompaktno rešitev.

Procesiranje kompleksnih pogojev predstavlja glavno funkcionalnost naše nadgradnje, zato ima ta knjižnica največjo vlogo. Skrbi za dodajanje, urejanje in brisanje CEP pravil iz vtičnika, zaznavanje dogodka, ki se ujame v eno izmed pravil in pošiljanje le tega v namenjen podatkovni model na platformi OM2M.

4.1.2 Jetty

Jetty [24] je odprtokodni spletni strežnik, ki omogoča deljenje statične ali dinamične vsebine in se lahko implementira v naprave, orodja, platforme in

aplikacijske strežnike. Knjižnica je izdana pod dvema licencama: Apache License 2.0 [42] in Eclipse Public License verzije 1.0 (EPL 1.0) [43].

Strežnik zavzema zelo malo pomnilniškega prostora, kar mu omogoča dobro performanco, skalabilnost in preprosto konfiguracijo (Slika 4.2). Vključitev knjižnice v aplikacijo omogoča, da se spletni strežnik uporablja kar znotraj aplikacije. Tak način uporabe je zelo priročen za manjše spletne aplikacije z nezahtevno vsebino, kar pa je primerno za uporabo v našem vtičniku.

Product	Vendor	Edition	Last release	Java EE compatibility [3]	Servlet	JSP	HTTP/2	License
ColdFusion	Adobe Systems	10.0.11	2013-07-09	6	2.5	2.1	No	Proprietary
Dynamo AS	ATG	6.3	2005-03	1.3	2.3	1.2	No	Proprietary
Enhydra	Lutris	5.1.9	2005-03-23	No			No	GPL
Enterprise Server	Borland	6.7	2007-01	1.4	2.4	2.0	No	Proprietary
Geronimo	ASF	3.0.1	2013-05-28	6 Full Platform	3.0	2.2	No	Apache License
GlassFish	GlassFish Community	4.1	2014-09	7 Full Platform	3.1	2.3	No	CDDL, GPL + classpath exception
iPlanet Web Server	Oracle Corporation	7.0.21	2015-04	Yes[4]	2.5	2.1	No	Proprietary
JBoss Enterprise Application Platform	Red Hat	6.4.0.GA	2015-04	6 Full Platform	3.0	2.2	No	LGPL
Jetty	Eclipse Foundation	9.3.3	2015-08-27	7 (partial[5])	3.1	2.3	Yes	Apache 2.0, EPL
JEUS	TmaxSoft	8	2013-08	7 Full Platform	3.0	2.2	No	Proprietary
JOnAS	OW2 Consortium (formerly ObjectWeb)	5.3	2013-10-04	6 Web Profile	3.0	2.2	No	LGPL
JRun	Adobe Systems	4 updater 7	2007-11-06	1.3	2.3	1.2	No	Proprietary
NetWeaver Application Server	SAP AG	7.4	2013-01-11	6	2.5	2.1	No	Proprietary
Oracle Containers for J2EE	Oracle Corporation	10.1.3.5.0	2009-08	1.4	2.4	2.0	No	Proprietary
Orion Application Server	IronFlare	2.0.7	2006-03-09	1.3	2.3	1.2	No	Proprietary
Resin Servlet Container (Open Source)	Caucho Technology	4.0.36	2013-04-25	6 Web Profile [6]	3.0	2.2	No	GPL
Resin Professional Application Server	Caucho Technology	4.0.36	2013-04-25	6 Web Profile	3.0	2.2	No	proprietary
Siwpas	MechSoft	2.0.0	2011-07-29	yes[7]	3.0	2.2	No	Proprietary
Tomcat	ASF	8.0.26	2015-08-21	7 (partial)	3.1	2.3	No	Apache License v2
TomEE	ASF	1.7.2	2015-05	6 Web Profile	3.0	2.2	No	Apache License
WebLogic Server	Oracle Corporation (formerly BEA Systems)	12.2.1	2015-10-26	7 Full Platform	3.1	2.3	No	Proprietary
WebObjects	Apple Inc.	5.4.3	2008-09-15	partial[8]			No	Proprietary
IBM WebSphere Application Server	IBM	8.5.5.6	2014-08-26	6 & 7 Full Platform	3.1	2.3	No	Proprietary
IBM WebSphere Application Server Liberty Core	IBM	8.5.5.6	2012-06-15	6 & 7 Web Profile	3.1	2.3	No	Proprietary
WebSphere AS Community Edition	IBM	3.0.0.4	2013-06-21	6 Full Platform	3.0	2.2	No	Proprietary
WildFly (fka JBoss AS)	Red Hat (formerly JBoss)	10	2016-01-29	7 Full Platform	3.1	2.3	Yes	LGPL

Slika 4.2: Primerjava Jetty z ostalimi spletnimi strežniki [25].

V našem vtičniku knjižnica omogoča prikaz in izbiro povezanih naprav, pregled, dodajanje, urejanje in brisanje CEP pravil preko grafičnega vmesnika ter avtentikacijo in vzdrževanje seje uporabnikov na spletni aplikaciji. Vse te funkcionalnosti so uporabniku na voljo v času delovanja sistema.

4.1.3 H2 Database Engine

H2 Database Engine [26] je odprtokodna relacijska podatkovna baza, izdana pod dvema licencama: Mozilla Public License 2.0 (MPL 2.0) [44] in Eclipse Public License verzije 1.0 (EPL 1.0) [43]. Podatkovna baza je zelo majhna in omogoča implementacijo v hitrem spominu, zato je primerna za uporabo v manjših aplikacijah, kjer poizvedbe na bazo niso zahtevne.

V primerjavi z drugimi podatkovnimi bazami je H2 Database Engine zelo majhna, za izdelavo podatkovne baze kreira le nekaj datotek, omogoča kompresijo BLOB objektov, vendar pa ne podpira procesiranje poizvedb na več nitih (Slika 4.3).

V nadgradnji platforme OM2M z rešitvijo za procesiranje kompleksnih dogodkov se s pomočjo knjižnice H2 hranijo, dodajajo, urejajo ter brišejo podatki o povezanih napravah na platformo in CEP pravilih.

Feature	H2	Derby	HSQLDB	MySQL	PostgreSQL
Pure Java	Yes	Yes	Yes	No	No
Embedded Mode (Java)	Yes	Yes	Yes	No	No
In-Memory Mode	Yes	Yes	Yes	No	No
Explain Plan	Yes	Yes *12	Yes	Yes	Yes
Built-in Clustering / Replication	Yes	Yes	No	Yes	Yes
Encrypted Database	Yes	Yes *10	Yes *10	No	No
Linked Tables	Yes	No	Partially *1	Partially *2	No
ODBC Driver	Yes	No	No	Yes	Yes
Fulltext Search	Yes	Yes	No	Yes	Yes
Domains (User-Defined Types)	Yes	No	Yes	Yes	Yes
Files per Database	Few	Many	Few	Many	Many
Row Level Locking	Yes *9	Yes	Yes *9	Yes	Yes
Multi Version Concurrency	Yes	No	Yes	Yes	Yes
Multi-Threaded Statement Processing	No *11	Yes	Yes	Yes	Yes
Role Based Security	Yes	Yes *3	Yes	Yes	Yes
Updatable Result Sets	Yes	Yes *7	Yes	Yes	Yes
Sequences	Yes	Yes	Yes	No	Yes
Limit and Offset	Yes	Yes *13	Yes	Yes	Yes
Window Functions	No *15	No *15	No	No	Yes
Temporary Tables	Yes	Yes *4	Yes	Yes	Yes
Information Schema	Yes	No *8	Yes	Yes	Yes
Computed Columns	Yes	Yes	Yes	No	Yes *6
Case Insensitive Columns	Yes	Yes *14	Yes	Yes	Yes *6
Custom Aggregate Functions	Yes	No	Yes	Yes	Yes
CLOB/BLOB Compression	Yes	No	No	No	Yes
Footprint (jar/dll size)	~1.5 MB *5	~3 MB	~1.5 MB	~4 MB	~6 MB

Slika 4.3: Primerjava H2 Database Engine z ostalimi podatkovnimi bazami [27].

4.2 Tehnične in vsebinske zahteve

Tehnične zahteve ter omejitve:

- Razširitev brez posega v obstoječo platformo OM2M
- Možnost vključitve nadgradnje kot vtičnik
- Možnost vključitve nadgradnje kot javanska knjižnica
- Čim večja svoboda razvijalcev pri uporabi nadgradnje
- Uporaba izključno odprtokodnih knjižnic.

Vsebinske zahteve:

- Možnost kreiranja neomejeno število CEP pravil na isto povezano napravo
- Pregled, dodajanje, urejanje in brisanje CEP pravil preko grafičnega vmesnika
- Dodajanje, urejanje in brisanje CEP pravil programsko
- Možnost naročitve naprav na dogodke, ujete v CEP pravilo
- Izpisovanje korakov za lažje razhroščevanje.

Sama platforma OM2M je implementirana kot »Plug-in« projekt v orodju Eclipse. To pomeni, da je sestavljena iz več vtičnikov in je lahko razširljiva. CEP nadgradnjo smo implementirali kot dodaten vtičnik, ki smo ga izvozili tudi kot javansko knjižnico, imenovano »om2m-cep.jar«. Nadgradnja ni del platforme in ne posega v obstoječo platformo, vendar pa uporablja razrede, implementirane v njej, kar pa pomeni, da je uporaba možna le v platformi OM2M.

Nadgradnja določa podatkovni tip sporočil le z vmesnikom, katerega razvijalec implementira v svojo OM2M platformo, ne omejuje kreiranje CEP

pravil z novo sintakso, vendar uporablja obstoječo, ki se uporablja tudi v knjižnici Esper. Knjižnica se prilagaja tudi na povezane naprave s platformo in s tem omogoča procesiranje kompleksnih dogodkov le z naprav, ki dejansko obstajajo. S tem smo želeli čim manj omejevati uporabnike in jim zagotoviti večjo svobodo pri uporabi.

Razviti vtičnik uporablja tri odprtokodne knjižnice za njegovo delovanje. Knjižnica Esper je glavna knjižnica vtičnika in zagotavlja procesiranje kompleksnih dogodkov, sintakso CEP pravil ter proženje dogodka ob uresničitvi enega od postavljenih pravil. Implementacijo spletnega vmesnika za pregled, dodajanje, urejanje in brisanje CEP pravil omogoča knjižnica Jetty, knjižnica H2 Database Engine pa omogoča shranjevanje povezanih naprav in CEP pravil v podatkovno bazo, implementirano v hitrem spominu.

Ob vsakokratnem kreiranju novega CEP pravila se v podatkovno bazo H2 doda nov zapis ter inicializira nov objekt, ki je klican ob uresničitvi kreiranega pravila. Omejitve na številu kreiranih pravil ni, obstaja le zavedanje, da ob večanju števila CEP pravil linearno narašča tudi število zapisov v bazi skupaj s številom inicializiranih objektov.

Vsak objekt na CEP vtičniku je povezan s svojim podatkovnim modelom na platformi OM2M pod določeno napravo. V primeru, da se dogodek ujame v pravilo, se posreduje na povezani podatkovni model, kjer se shrani in je na voljo za pregled pod okriljem platforme OM2M. S tem je omogočeno tudi naročanje na dogodke, ki se ujamejo v določeno pravilo, saj sama platforma omogoča tudi mehanizem objavi/naroči na svojih podatkovnih modelih.

Zaradi optimalnega delovanja vtičnik v času izvajanja privzeto ne beleži veliko informacij, zato je bila dodana možnost vklopa in izklopa izpisovanja korakov, ki se izvajajo v nadgradnji. S tem smo omogočili lažje sledenje in iskanje napak v samem postavljanju platforme OM2M.

4.3 Delovanje nadgradnje

Ob zagonu platforme OM2M, skupaj z nadgradnjo za procesiranje kompleksnih dogodkov, se poleg spletnega strežnika platforme zažene tudi spletni strežnik nadgradnje.

Spletni strežnik platforme OM2M ponuja grafični vmesnik za prikaz drevesne strukture platforme OM2M, prikaz naprav in podatkov, ki se pošiljajo iz le teh ter možnost pošiljanja prej definiranih http zahtev na samo platformo. Dostopen je preko vrat 8080, preko url naslova http://localhost:8080/webpage/ in je zavarovan z uporabniškim imenom in geslom (privzeto admin:admin).

Strežnik nadgradnje ponuja grafični vmesnik za prikaz, kreiranje, urejanje in brisanje CEP pravil. Dostopen je preko vrat 8081, in sicer preko url naslova http://localhost:8081/cep/. Dostop do spletnega vmesnika je zavarovan z uporabniškim imenom in geslom, ki pa sta zaradi preprostosti vedno enaka kot na spletnem vmesniku platforme OM2M.

Za uspešno dodajanje novega pravila za procesiranje kompleksnih dogodkov je potrebno izbrati povezano napravo, na kateri se bo postavilo novo pravilo, izbrati ime podatkovnega modela, v katerega se bodo ulovljeni dogodki zapisovali ter vpisati pravilo v sintaksi Esper EPL brez napak. Po potrditvi se na strani nadgradnje v podatkovno bazo zapišejo vsi podatki o pravilu ter kreira nova instanca za lovljenje dogodkov, ki se bodo ujeli v zastavljeno pravilo. Na strani platforme OM2M pa se kreira nov podatkovni model z izbranim imenom, v katerega se pošiljajo in shranjujejo dogodki, ujeti na strani nadgradnje.

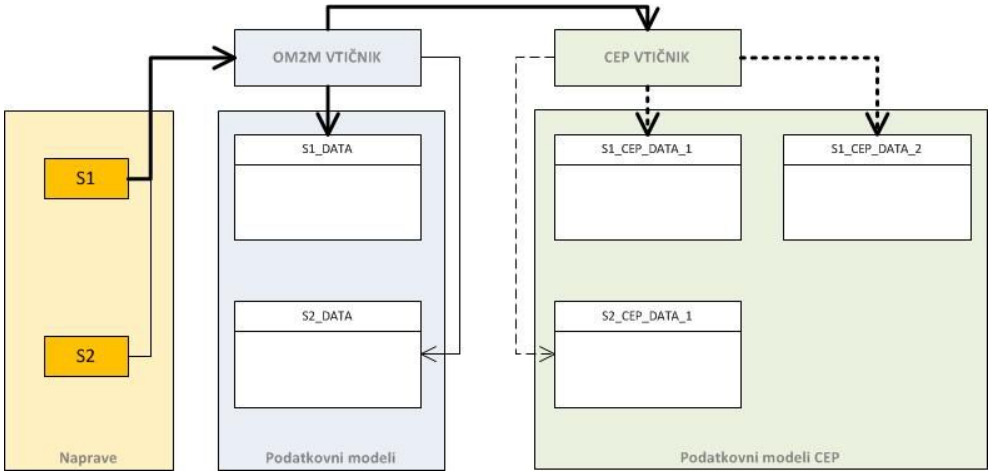
Spodnja slika prikazuje podatkovni tok platforme OM2M z implementirano nadgradnjo za procesiranje kompleksnih dogodkov. V primeru sta na platformo povezani dve napravi (S1 in S2) ter implementirana dva podatkovna modela (`S1_DATA` in `S2_DATA`), kjer se bodo prejeti podatki iz teh dveh naprav shranjevali. Platforma vsebuje tudi dva podatkovna modela za shranjevanje dogodkov (`S1_CEP_DATA_1`, `S1_CEP_DATA_2` ter `S2_CEP_DATA_1`), ki se bodo ujeli v prej definirana CEP pravila.

- `S1_CEP_DATA_1` – Podatkovni model za kreirano CEP pravilo 1, za napravo S1

- `S1_CEP_DATA_2` – Podatkovni model za kreirano CEP pravilo 2, za napravo S1

- `S2_CEP_DATA_1` – Podatkovni model za kreirano CEP pravilo 1, za napravo S2

Na sliki 4.4 odebeljena pot prikazuje podatkovni tok, ko naprava S1 pošlje podatek na platformo OM2M. Vtičnik na platformi vedno shrani podatek v napravi namenjeni podatkovni model (`S1_DATA`) ter ga posreduje nadgradnji za procesiranje kompleksnih dogodkov. Nadgradnja preveri, če se podatek ujame v katero koli pravilo, ki je namenjeno napravi S1 in po potrebi proži zapis dogodka v temu namenjen podatkovni model (`S1_CEP_DATA_1` ali `S1_CEP_DATA_2`).



Slika 4.4: Delovanje nadgradnje za procesiranje kompleksnih dogodkov.

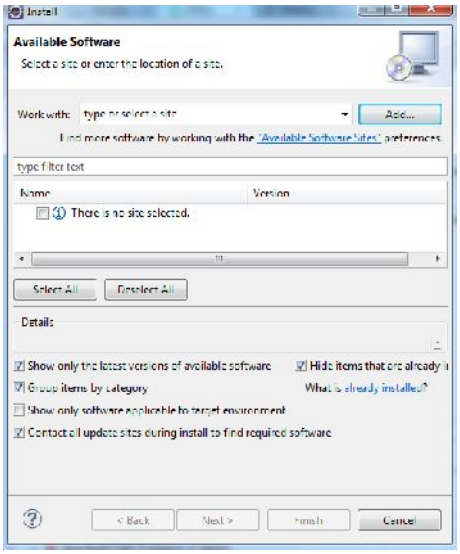
4.4 Vzpostavitev platforme OM2M z nadgradnjo za procesiranje kompleksnih dogodkov

Platforma OM2M je razvita kot plug-in projekt v orodju Eclipse, zato je za vzpostavitev platforme potrebno imeti orodje Eclipse za RCP in RAP razvijalce [45].

4.4.1 Uvoz platforme OM2M

Konfigurator Tycho

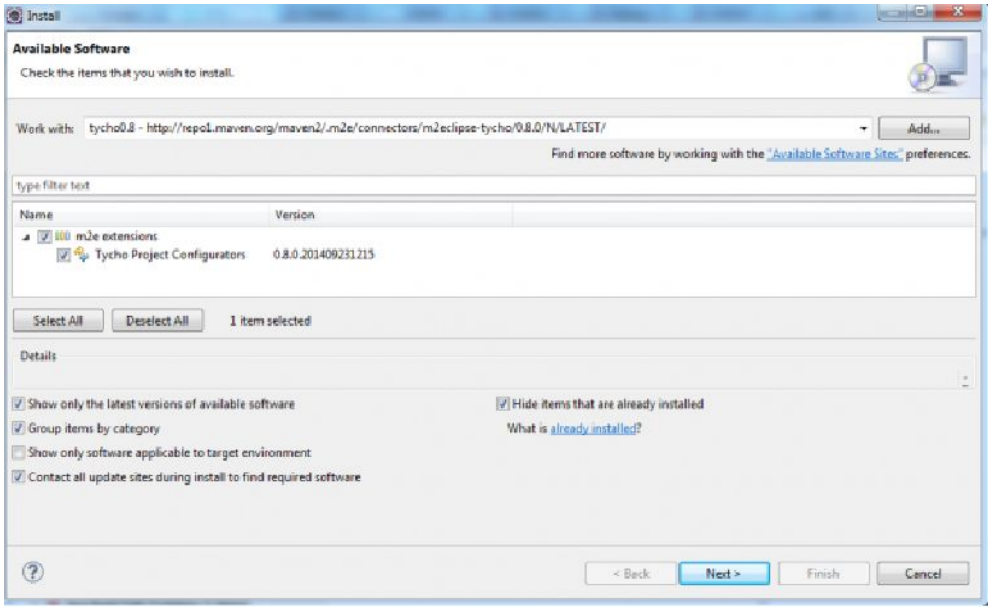
Za izdelavo plug-in projekta v okolju Eclipse potrebujemo Tycho konfigurator [46], ki ga naložimo preko repozitorija za posodobitve. Odpremo orodje Eclipse, v meniju označimo »Help« in izberemo »Install New Software«. Pritisnemo na gumb »Add...«, ki nam odpre okno za dodajanje repozitorijev (Slika 4.5).



Slika 4.5: Inštalacija novih aplikacij.

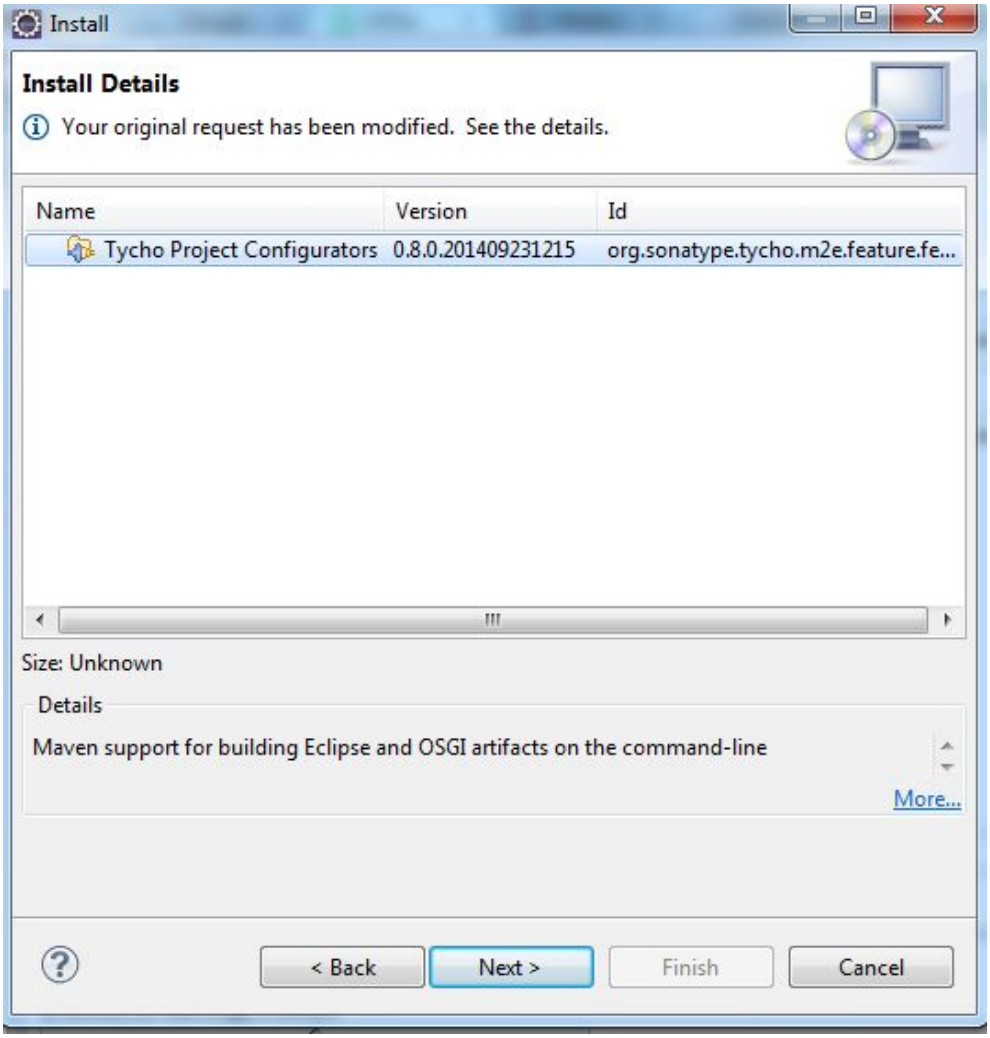
V polje »Name« vpišemo »tycho0.8«, v polje »Location« pa »http://repo1.maven.org/maven2/.m2e/connectors/m2eclipse-tycho/0.8.0/N/LATEST/« in pritisnemo na gumb »OK«.

Obkljukamo vrstico »m2e extensions« in »Tycho Project Configurators« ter pritisnemo na gumb »Next« (Slika 4.6).



Slika 4.6: Dodajanje Tycho repozitorija.

Izberemo »Tycho Project Configurators« in pritisnemo na gumb »Next« (Slika 4.7).

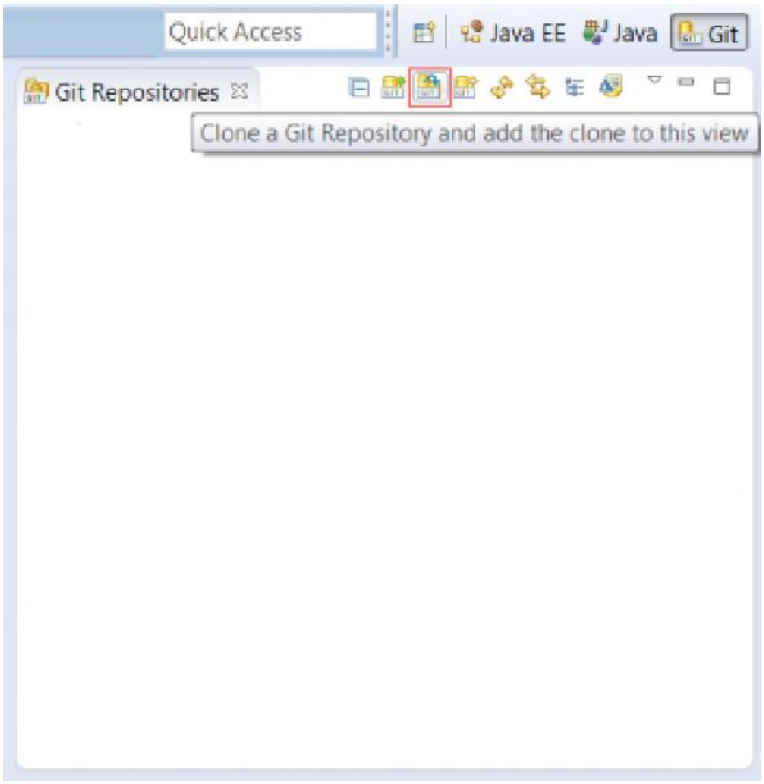


Slika 4.7: Inštalacija aplikacije Tycho.

V zadnjem koraku dodajanja Tycho konfiguratorja sprejmemo pogoje uporabe in kliknemo na gumb »Finish«.

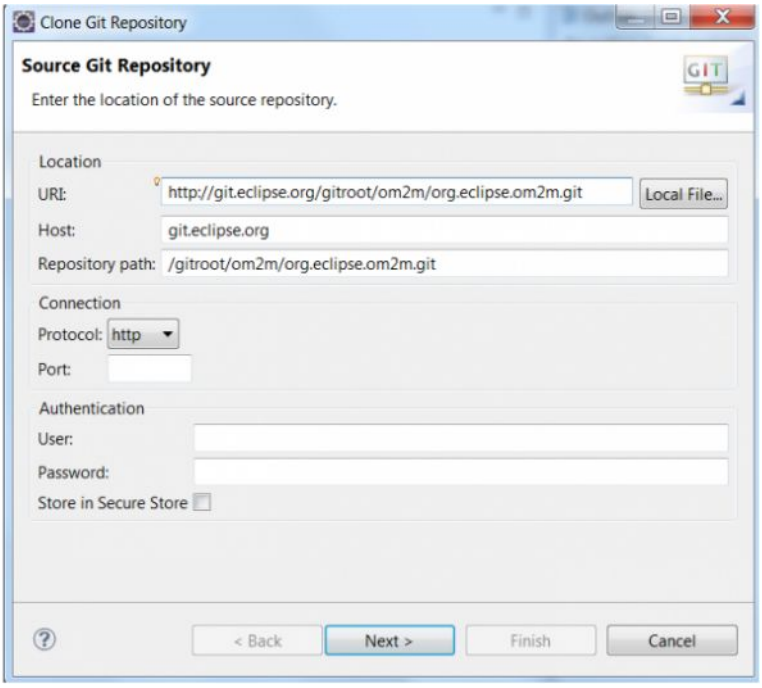
Kloniranje platforme OM2M

Po uspešni namestitvi aplikacije Tycho, v meniju orodja Eclipse označimo »Window«, nato »Show View« in izberemo »Other«. V oknu, ki se pokaže, izberemo »Git« pogled in kliknemo na ikono za kloniranje Git repozitorija (Slika 4.8).



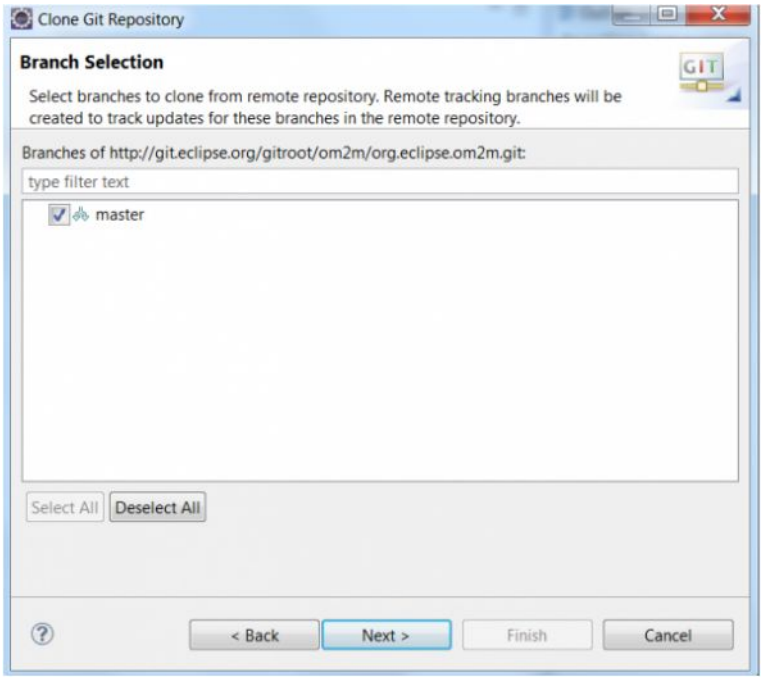
Slika 4.8: Kloniranje OM2M repozitorija git v orodje Eclipse.

Kloniramo OM2M projekt z uporabo url naslova »http://git.eclipse.org/gitroot/om2m/org.eclipse.om2m.git«, ki ga vpišemo v polje »URI« in izberemo »Next« (Slika 4.9).



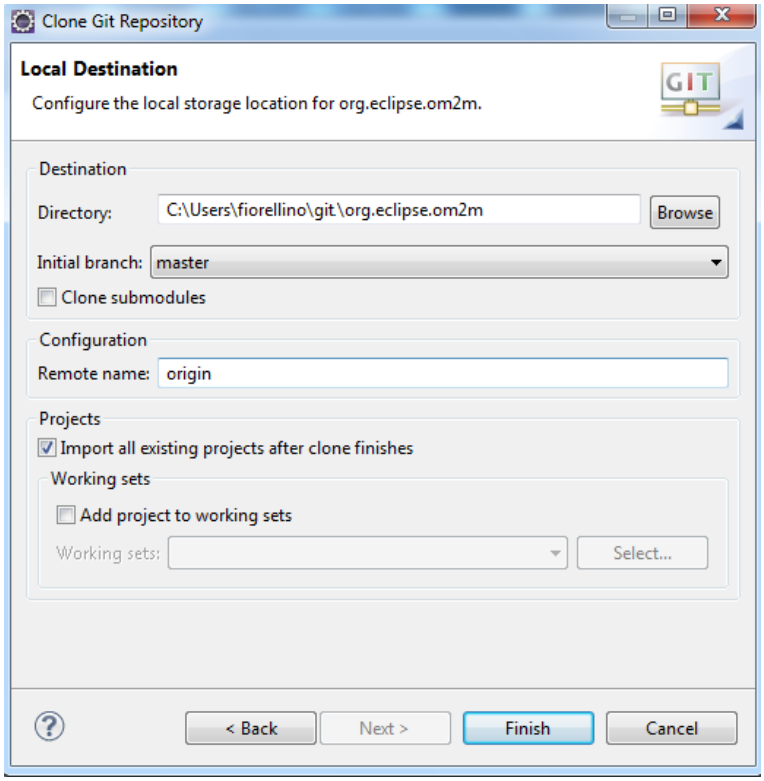
Slika 4.9: Urejanje povezave z repozitorijem git platforme OM2M.

Na naslednji strani moramo imeti označeno vejo repozitorija »master« (Slika 4.10).



Slika 4.10: Master veja repozitorija OM2M.

Na tretji strani izberemo polje »Import all existing projects after clone finishes« in pritisnemo na gumb »Finish« (Slika 4.11).



Slika 4.11: Vključitev projekta po uspešnem kloniranju.

Projekt OM2M je uspešno kloniran in je viden v oknih »Git repositories« in »Package explorer«.

Zagon in test platforme

Izberemo »org.eclipse.om2m« paket z desnim klikom. Označimo možnost »Run as« in izberemo »Maven install«. Ko se platforma uspešno zgradi, bodo na voljo trije produkti orodja Eclipse:

- IN-CSE produkt, ki ga je mogoče najti v imeniku `om2m/org.eclipse.om2m/org.eclipse.om2m.site.in-cse/target/products/in-cse/<os>/<ws>/<arch>`

- MN-CSE produkt, ki ga je mogoče najti v imeniku `om2m/org.eclipse.om2m/org.eclipse.om2m.site.mn-cse/target/products/mn-cse/<os>/<ws>/<arch>`
- ASN-CSE produkt, ki ga je mogoče najti v imeniku `om2m/org.eclipse.om2m/org.eclipse.om2m.site.asn-cse/target/products/asn-cse/<os>/<ws>/<arch>`

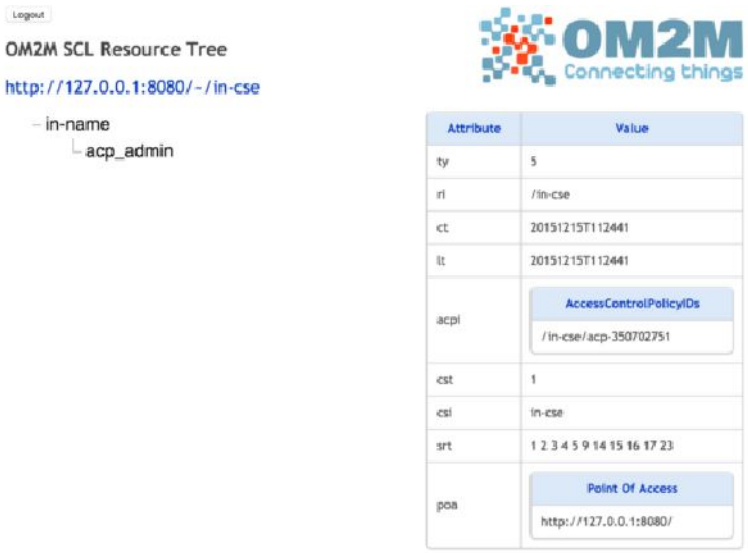
Odpremo orodje za brskanje po datotečnem sistemu in obiščemo imenik IN-CSE produkta `om2m/org.eclipse.om2m/org.eclipse.om2m.site.in-cse/target/products/in-cse/<os>/<ws>/<arch>`. Na operacijskem sistemu Windows zaženemo datoteko »start.bat«, na operacijskem sistemu Linux in Mac pa »start.sh«. Ko je produkt IN-CSE zagnan, vidimo konzolo OSGi. Vpišemo »ss« za pregled statusov vseh naloženih vtičnikov kot prikazuje slika 4.12.

Slika 4.12: Statusi vseh naloženih vtičnikov.

Odpremo spletni brskalnik, kjer za dostop do spletnega vmesnika v url vpišemo naslov »http://127.0.0.1:8080/webpage« (Slika 4.13). Vpišemo privzeto uporabniško ime in geslo »admin«, »admin« in pritisnemo gumb »Login« za prikaz drevesne strukture produkta.

Slika 4.13: Vpis v spletno mesto platforme OM2M.

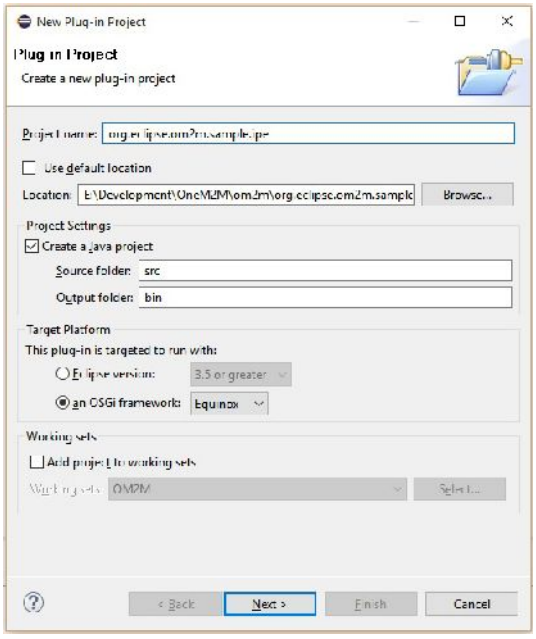
Po uspešni avtentikaciji bo izpisana drevesna struktura produkta IN-CSE (Slika 4.14).



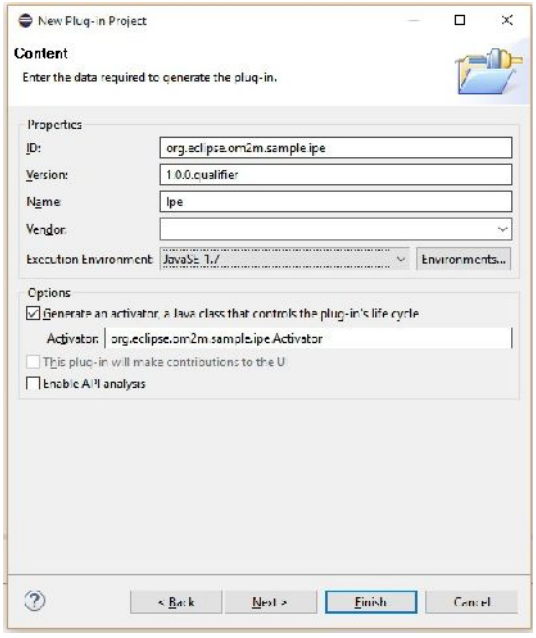
Slika 4.14: Drevesna struktura produkta IN-CSE.

4.4.2 Kreiranje lastnega vtičnika

Predpostavljamo, da je platforma OM2M že uspešno uvožena v orodje Eclipse. V meniju označimo »File«, nato »New«, potem »Other« in izberemo »Plug-in Project«. Vpišemo podatke kot so prikazani na sliki 4.15 in sliki 4.16.

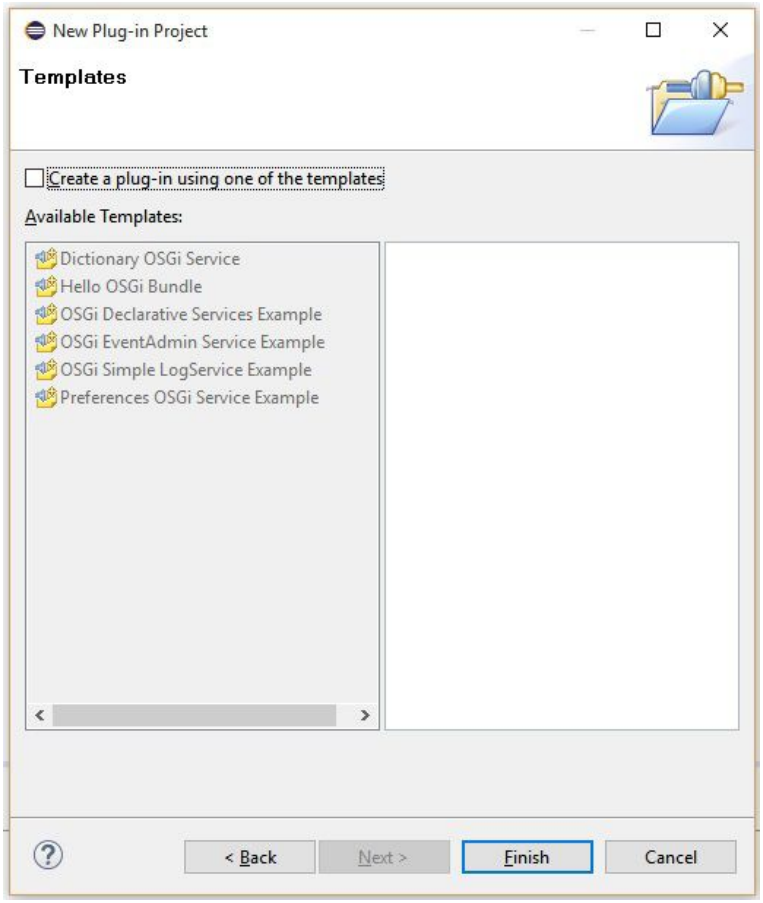


Slika 4.15: Kreiranje novega vtičnika v orodju Eclipse.



Slika 4.16: Kreiranje novega vtičnika v orodju Eclipse (drugi del).

Odznačimo oznako »Create a plug-in using one of the templates« in pritisnemo na gumb »Finish« (Slika 4.17).

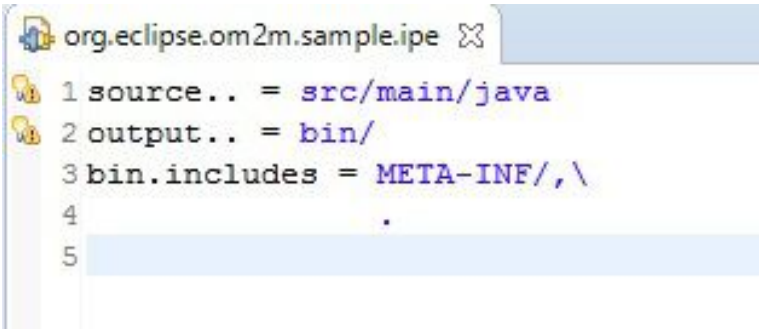


Slika 4.17: Predloge kreiranega vtičnika.

Rezultat je kreiran vtičnik org.eclipse.om2m.sample.ipe (Slika 4.18).

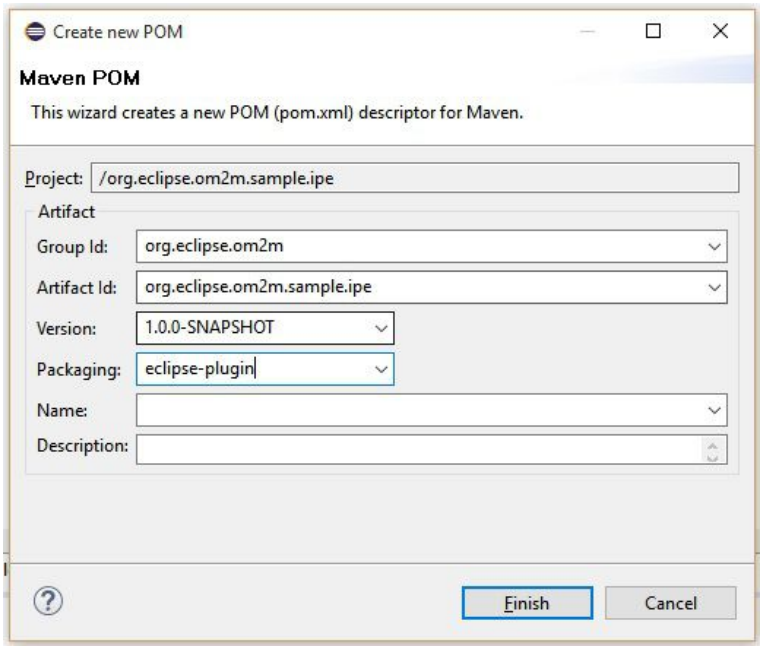
Slika 4.18: Drevesna struktura kreiranega vtičnika.

Odpremo datoteko »build.properties« in popravimo vrednosti tako kot so prikazane na sliki 4.19.



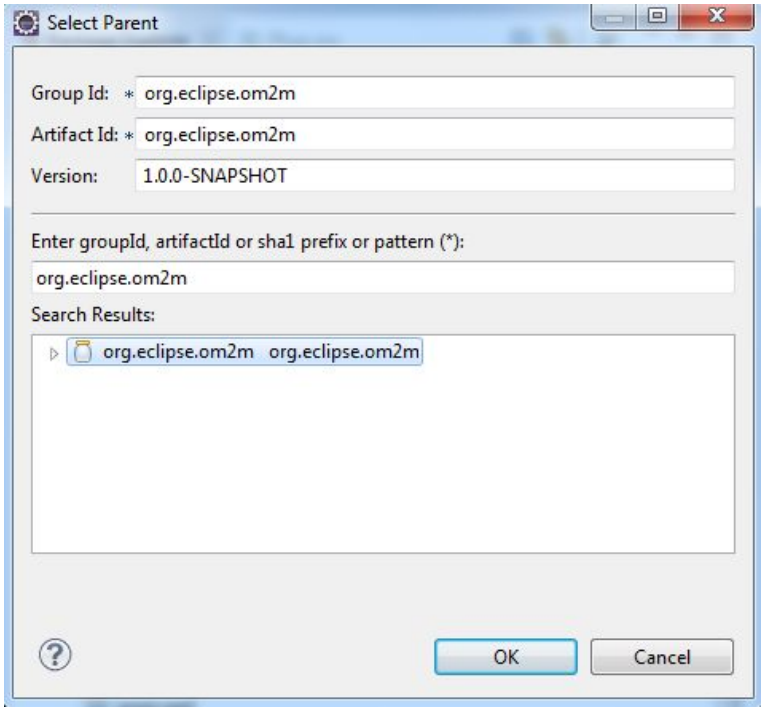
Slika 4.19: Ureditev datoteke "build.properties".

Ustvarjeni vtičnik org.eclipse.om2m.sample.ipe kliknemo z desnim miškinim gumbom, označimo »Configure« in kliknemo na »Convert to maven project«. Vpišemo podatke kot so na sliki 4.20 in pritisnemo na gumb »Finish«.



Slika 4.20: Menjava tipa projekta v Maven projekt.

Ko je vtičnik pretvorjen v maven projekt, odpremo datoteko »pom.xml«. Kliknemo na ikono »Select Parent« in v polja »Group Id« in »Artifact Id« vnesemo »org.eclipse.om2m«. Iz rezultatov izberemo »org.om2m.eclipse« in pritisnemo na gumb »OK« (Slika 4.21).



Slika 4.21: Izbira nadrejenega vtičnika.

Nastavitve zaključimo s posodobitvijo kreiranega vtičnika. Z desnim gumbom izberemo projekt »org.eclipse.om2m.sample.ipe«, označimo »Maven« in izberemo »Update project«.

Vtičnik je sedaj potrebno dodati kot modul v starševski projekt OM2M. V projektu »org.eclipse.om2m« odpremo datoteko »pom.xml«. Izberemo zavihek »Modules« in pritisnemo na gumb »Add«. Izberemo vtičnik »org.eclipse.om2m.sample.ipe« in pritisnemo na gumb »OK«. Z desnim klikom izberemo projekt »org.eclipse.om2m«, označimo možnost »Run as« in pritisnemo na »Maven install«. Ko se projekt uspešno zgradi, bi v konzoli morali dobiti rezultat kot je prikazan na sliki 4.22.

```
[INFO] org.eclipse.om2m.sample.ipe ..................... SUCCESS [  0.055 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 37.614 s
[INFO] Finished at: 2015-11-12T11:24:59+01:00
[INFO] Final Memory: 99M/511M
[INFO] ------------------------------------------------------------------------
```

Slika 4.22: Rezultat izgradnje platforme OM2M.

V naslednjem koraku bomo kreirani vtičnik dodali v enega izmed OM2M produktov, kar pomeni, da se bo vtičnik lahko zagnal iz izbranega produkta (in-cse, mn-cse, asn-cse). Izberemo produktni vtičnik, na primer org.eclipse.om2m.site.in-cse, in odpremo njegovo datoteko »om2m.product«. Izberemo zavihek »Dependencies« in kliknemo na gumb »Add«. Vpišemo »org.eclipse.om2m.sample.ipe«, izberemo vtičnik in potrdimo s klikom na gumb »OK« in »Save«. Zgradimo platformo OM2M tako, da z desnim klikom izberemo projekt »org.eclipse.om2m«, izberemo »Run As« in kliknemo na možnost »Maven install«.

Da je bil vtičnik »org.eclipse.om2m.sample.ipe« uspešno dodan v OM2M produkt IN-CSE, zaženemo datoteko »start.bat« na lokaciji produkta. Ko se produkt uspešno zažene, v terminal vpišemo »ss« in preverimo, če izpis vsebuje vtičnik »org.eclipse.om2m.sample.ipe«, kot je prikazano na sliki 4.23.

```
osgi> ss
"Framework is launched."

id      State       Bundle
0       ACTIVE      org.eclipse.osgi_3.10.2.v20150203-1939
1       RESOLVED    javax.servlet_3.1.0.v20140303-1611
2       RESOLVED    javax.xml_1.3.4.v201005080400
3       RESOLVED    org.apache.commons.codec_1.6.0.v201305230611
4       RESOLVED    org.apache.commons.logging_1.1.1.v201101211721
                    Fragments=24
5       ACTIVE      org.apache.felix.gogo.command_0.10.0.v201209301215
6       ACTIVE      org.apache.felix.gogo.runtime_0.10.0.v201209301036
7       ACTIVE      org.apache.felix.gogo.shell_0.10.0.v201212101605
8       RESOLVED    org.apache.httpcomponents.httpclient_4.3.6.v201411290715
9       RESOLVED    org.apache.httpcomponents.httpcore_4.3.3.v201411290715
10      ACTIVE      org.eclipse.equinox.console_1.1.0.v20140131-1639
11      ACTIVE      org.eclipse.equinox.http.jetty_3.0.200.v20131021-1843
12      ACTIVE      org.eclipse.equinox.http.servlet_1.1.500.v20140318-1755
13      RESOLVED    org.eclipse.equinox.launcher_1.3.0.v20140415-2008
14      RESOLVED    org.eclipse.jetty.continuation_8.1.16.v20140903
15      RESOLVED    org.eclipse.jetty.http_8.1.16.v20140903
16      RESOLVED    org.eclipse.jetty.io_8.1.16.v20140903
17      RESOLVED    org.eclipse.jetty.security_8.1.16.v20140903
18      RESOLVED    org.eclipse.jetty.server_8.1.16.v20140903
19      RESOLVED    org.eclipse.jetty.servlet_8.1.16.v20140903
20      RESOLVED    org.eclipse.jetty.util_8.1.16.v20140903
21      ACTIVE      org.eclipse.om2m.binding.http_1.0.0.20160322-0626
22      RESOLVED    org.eclipse.om2m.binding.service_1.0.0.20160322-0626
23      RESOLVED    org.eclipse.om2m.commons_1.0.0.20160322-0626
24      RESOLVED    org.eclipse.om2m.commons.logging_1.0.0.20160322-0626
                    Master=4
25      ACTIVE      org.eclipse.om2m.core_1.0.0.20160322-0626
26      RESOLVED    org.eclipse.om2m.core.service_1.0.0.20160322-0626
27      ACTIVE      org.eclipse.om2m.datamapping.jaxb_1.0.0.20160322-0626
28      RESOLVED    org.eclipse.om2m.datamapping.service_1.0.0.20160322-0626
29      RESOLVED    org.eclipse.om2m.interworking.service_1.0.0.20160322-0626
30      ACTIVE      org.eclipse.om2m.persistence.eclipselink_1.0.0.20160322-0626

31      RESOLVED    org.eclipse.om2m.persistence.service_1.0.0.20160322-0626
32      STARTING    org.eclipse.om2m.sample.ipe_1.0.0.20160322-0626
33      RESOLVED    org.eclipse.om2m.sample.sensor_1.0.0.20160322-0626
34      ACTIVE      org.eclipse.om2m.webapp.resourcesbrowser.xml_1.0.0.20160322-
0626
35      RESOLVED    org.eclipse.osgi.services_3.4.0.v20140312-2051
osgi>
```

Slika 4.23: Stanje kreiranega vtičnika v zagnani platformi OM2M.

Vtičnik je uspešno kreiran in dodan v platformo OM2M, vendar pa mu še nismo implementirali nobenih funkcionalnosti. Za primer procesiranja podatkov je v samo platformo že implementiran vtičnik »org.eclipse.om2m.ipe.sample«, ki prikazuje prižiganje luči s pomočjo uporabniškega vmesnika, preko platforme OM2M. Na uradni strani OM2M je opisan tudi primer vtičnika, ki procesira vrednosti povezanega senzorja [47].

Za vtičnik, na katerem bomo testirali našo knjižnico, je slednja implementacija bolj primerna, saj vsebuje le osnovne komponente in je tako bolj preprosta za pregled in uporabo.

Activator.java

Razred Activator vsebuje implementacijo metode za aktivacijo in deaktivacijo vtičnika (start in stop). Metoda start registrira storitev vtičnika v register, da je na voljo iz glavnega vtičnika, metoda stop pa počisti določene vire pred deaktivacijo vtičnika kot so ustavljanje niti, ustavljanje razreda Monitor, ravnanje z izjemami in podobno.

Monitor.java

Razred Monitor naredi senzor po imenu »`MY_SENSOR`«, tako imenovani aktuator »`MY_ACTUATOR`« ter dva modela za vsako aplikacijo »DESCRIPTOR«, ki vsebuje opis in »DATA«, ki shranjuje podatke prejete iz senzorja. Razred Monitor kreira tudi dve niti, ki spremljata modela in ob vsakem prejetem dogodku izdela zapis z meritvami. Na voljo je tudi metoda, ki ustavi razred Monitor in njegovi niti.

ObixUtil.java

Razred ObixUtil je namenjen pretvorbi opisa in vrednosti podatkov v oBIX XML obliko sporočila, ki se pošilja iz enega na drug produkt.

Controller.java

Razred Controller izvrši prejete HTTP zahteve iz platforme OM2M.

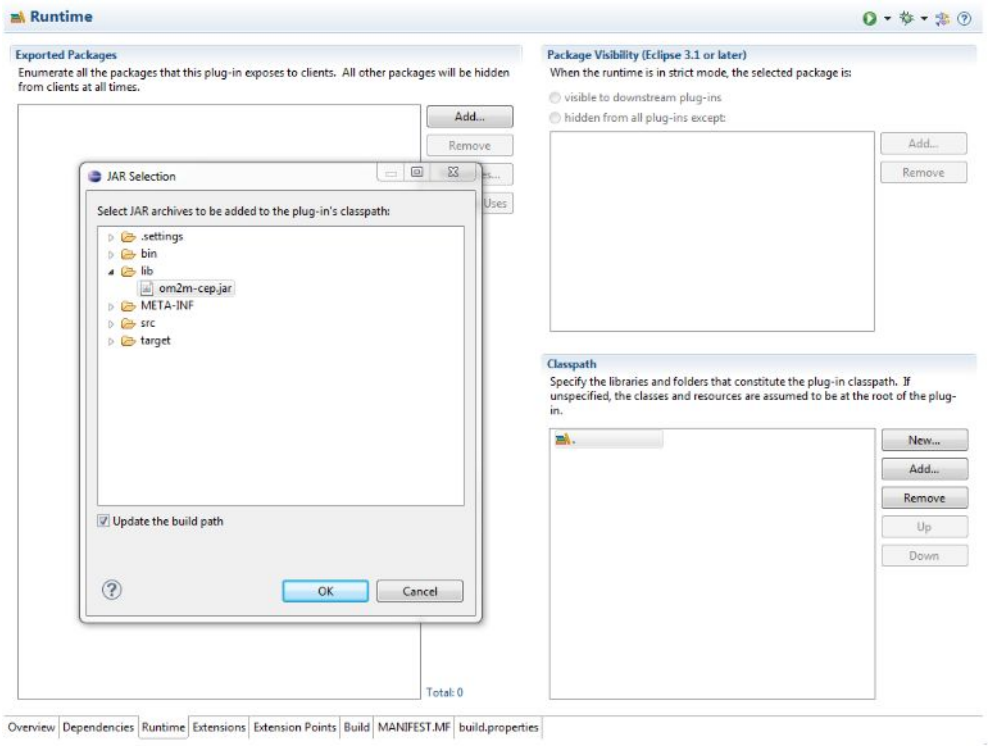
RequestSender.java

Razred RequestSender vsebuje metode, s katerimi poenostavlja pošiljanje zahtev na platformo.

4.4.3 Implementacija CEP nadgradnje

Vključitev knjižnice CEP v platformo OM2M

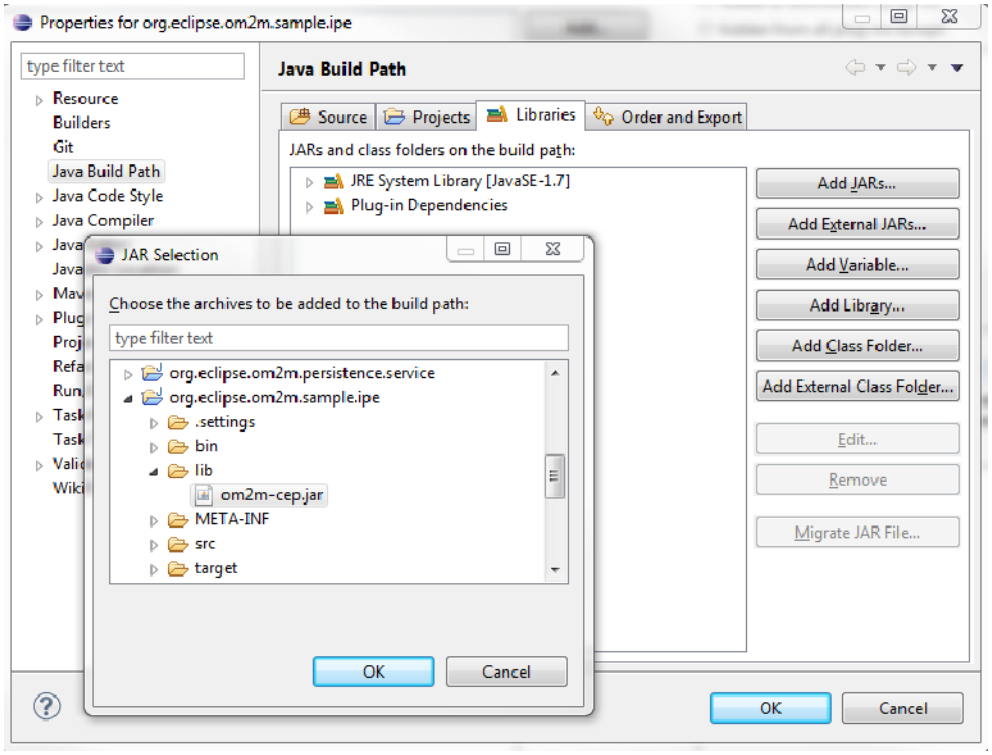
Iz repozitorija GitHub [48] prenesemo knjižnico »om2m-cep.jar«. V prej ustvarjenem vtičniku »org.eclipse.om2m.sample.ipe« kreiramo mapo z imenom »libs«, kjer odložimo knjižnico »om2m-cep.jar«. V orodju Eclipse odpremo datoteko »MANIFEST.MF« iz vtičnika »org.eclipse.om2m.sample.ipe«. Izberemo zavihek »Runtime« in v sekciji »Classpath« kliknemo na gumb »Add«. Izberemo knjižnico »om2m-cep.jar« znotraj mape »libs« in potrdimo s klikom na gumb »OK« (Slika 4.24).



Slika 4.24: Dodajanje knjižnice za procesiranje kompleksnih dogodkov v orodje Eclipse.

Zaradi razpoznavanja knjižnice »om2m-cep.jar« v orodju Eclipse jo je potrebno dodati tudi v sam projekt. Z desnim klikom označimo projekt

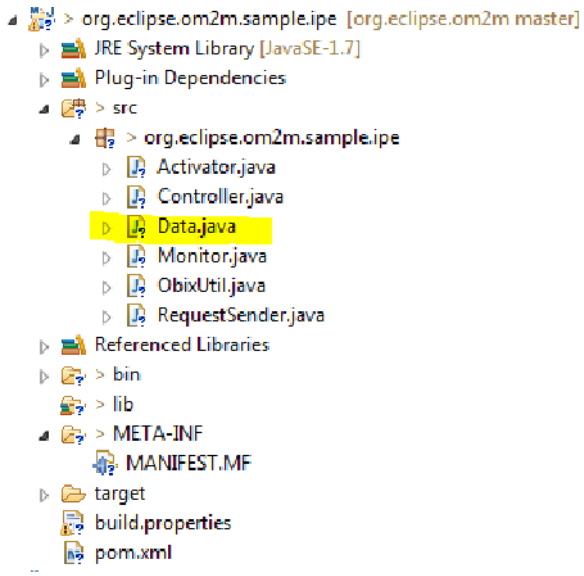
»org.eclipse.om2m.sample.ipe« in izberemo »Properties«. Pomaknemo se v sekcijo »Java Build Path«, zavihek »Libraries« in pritisnemo na gumb »Add JARs. . . «. Iz ponujenih vtičnikov odpremo »org.eclipse.om2m.sample.ipe« in izberemo knjižnico »om2m-cep.jar« iz mape »libs«. Okna potrdimo s klikom na gumb »OK« (Slika 4.25).



Slika 4.25: Dodajanje knjižnice za procesiranje kompleksnih dogodkov v projekt.

Primer uporabe

V vtičniku, ki smo ga naredili, naredimo nov razred, ki bo implementacija razreda DataInterface (Slika 4.26).



Slika 4.26: Implementacija razreda DataInterface.

V tem razredu je potrebno deklarirati podatke, ki se bodo pošiljali iz določene naprave in bodo uporabljeni za procesiranje kompleksnih pogojev. V našem primeru bomo procesirali samo eno število, imenovano »value«. Razred mora implementirati podatkovno strukturo DataInterface, ki pa zahteva konstruktor ter za vsako procesirano vrednost metodi get() in set() (Slika 4.27).

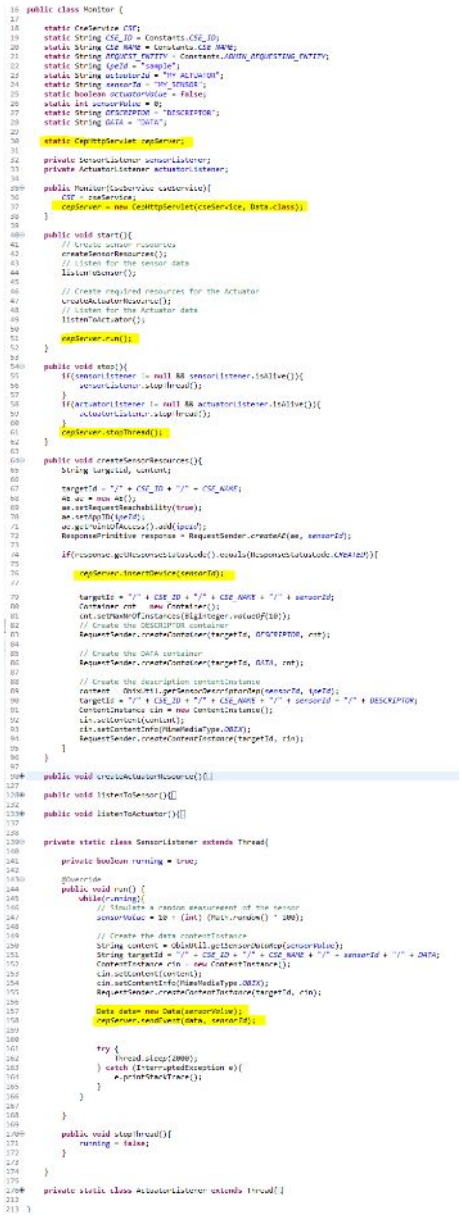
```
package org.eclipse.om2m.sample.ipe;

import si.fri.mag.gasperin.cep.utils.DataInterface;

public class Data implements DataInterface{
    double value;

    public Data(double value) {
        this.value = value;
    }

    public double getValue() {
        return value;
    }

    @Override
    public String toString() {
        return "" + value;
    }
}
```

Slika 4.27: Metode, ki implementirajo razred DataInterface.

V nadaljevanju moramo dodelati vtičnik platforme OM2M, saj je pri zagonu vtičnika potrebno inicializirati in zagnati instanco razreda za kompleksno procesiranje dogodkov, ji podati podatkovno strukturo, ki jo bo procesirala, dodati naprave, ki so povezane s platformo in nad katerimi bomo postavljali CEP pogoje ter v primeru prejetja podatka iz naprave le tega posredovati v nadgradnjo CEP. V primeru bomo vse to implementirali znotraj datoteke Monitor.class (Slika 4.28):

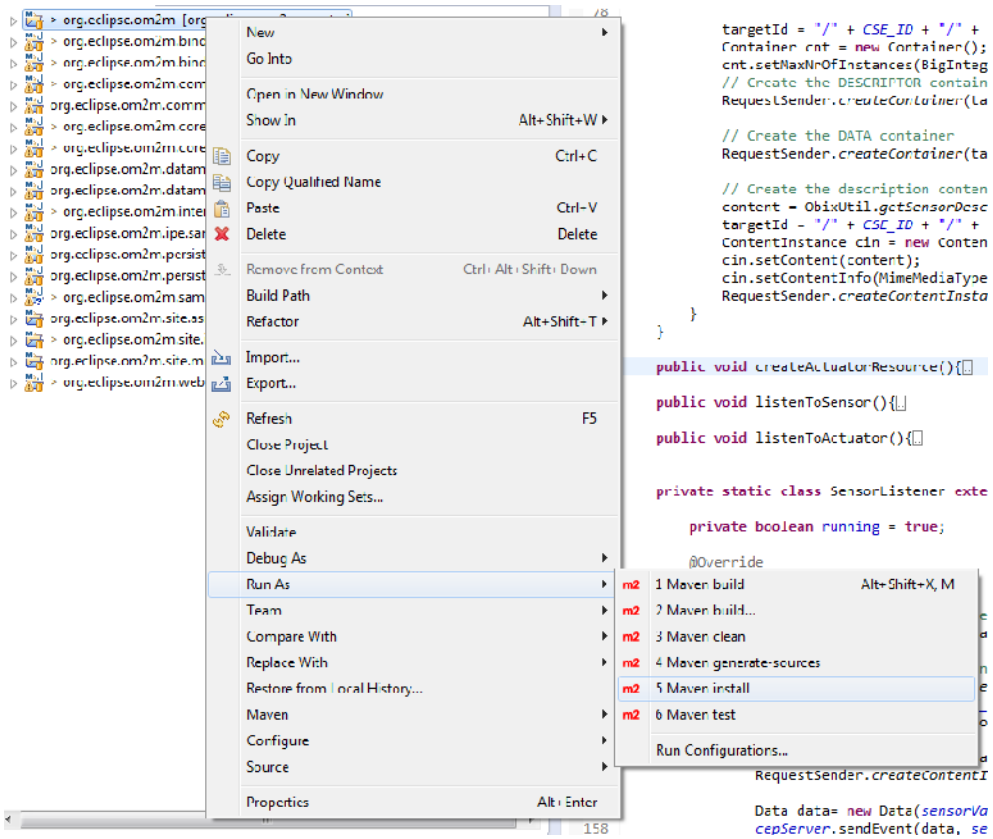
- Kreiramo instanco razreda CepHttpServlet: static CepHttpServlet cepServer;
- V konstruktorju razreda inicializiramo spremenljivko cepServer: cepServer = new CepHttpServlet(cseService, Data.class);
- V metodi start() zaženemo CEP strežnik: cepServer.run();
- V metodi stop() ustavimo CEP strežnik: cepServer.stopThread();
- V metodi createSensorResources() dodamo vse povezane naprave: cepServer.insertDevice(sensorId);

- V razredu SensorListener, v metodo run() pošljemo prejete podatke naprej v nadgradnjo CEP: Data data = new Data(sensorValue); cepServer.sendEvent(data, sensorId);



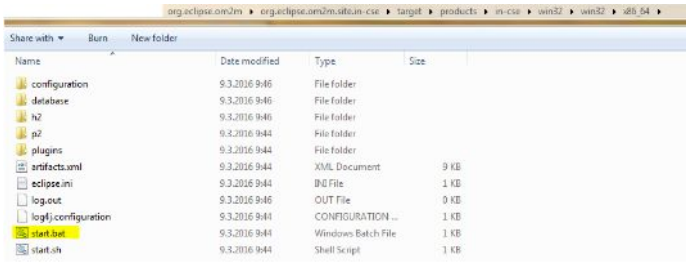
Slika 4.28: Modifikacija razreda Monitor.

V naslednjem koraku zgradimo in zaženemo celoten projekt OM2M (Slika 4.29).



Slika 4.29: Zagon platforme OM2M.

Navigiramo na IN-CSE produkt, ki ga je mogoče najti v imeniku `om2m/org.eclipse.om2m/org.eclipse.om2m.site.in-cse/target/products/in-cse/<os>/<ws>/<arch>` in odpremo datoteko »start.bat« (Slika 4.30).

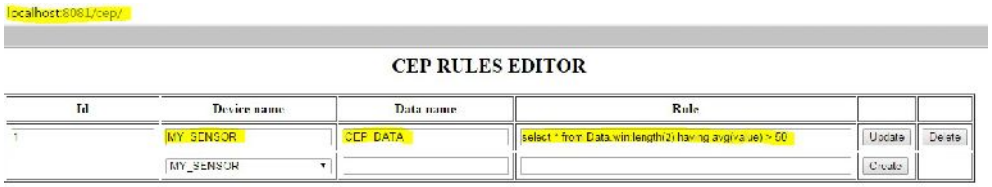


Slika 4.30: Imenik platforme OM2M in datoteka start.bat.

V terminal vpišemo »ss«, da poiščemo naš OM2M vtičnik, v katerega smo implementirali knjižnico CEP. Z ukazom »start id« zaženemo željen vtičnik, kjer je id enak idju iskanega vtičnika. V primeru kot ga prikazuje spodnja slika vpišemo »start 32« (Slika 4.31).

Slika 4.31: Zagon kreiranega vtičnika v platformi OM2M.

V spletnem brskalniku odpremo url naslov »http://localhost:8081/cep« in se avtenticiramo (privzete vrednosti so admin:admin). V polju »Device name« izberemo napravo, nad katero bomo postavljali CEP pravila. V našem primeru bo to »`MY_SENSOR`«. V polje »Data name« vpišemo ime podatkovnega modela, ki bo prikazan v drevesni strukturi platforme OM2M. V našem primeru vpišemo »`CEP_DATA`«. V polje »Rule« z sintakso Esper EPL vpišemo CEP pravilo za iskanje kompleksnih dogodkov. V našem primeru vpišemo »`select * from Data.win:length(2) having avg(value) > 50`«, kar pomeni, da CEP nadgradnja proži dogodek, ko prejme dve zaporedni vrednosti, poslani iz naprave »`MY_SENSOR`«, katerih povprečje je večje od 50. S pritiskom na gumb »Create« ustvarimo CEP pravilo (Slika 4.32).



Slika 4.32: Kreiranje pravila CEP (GUI).

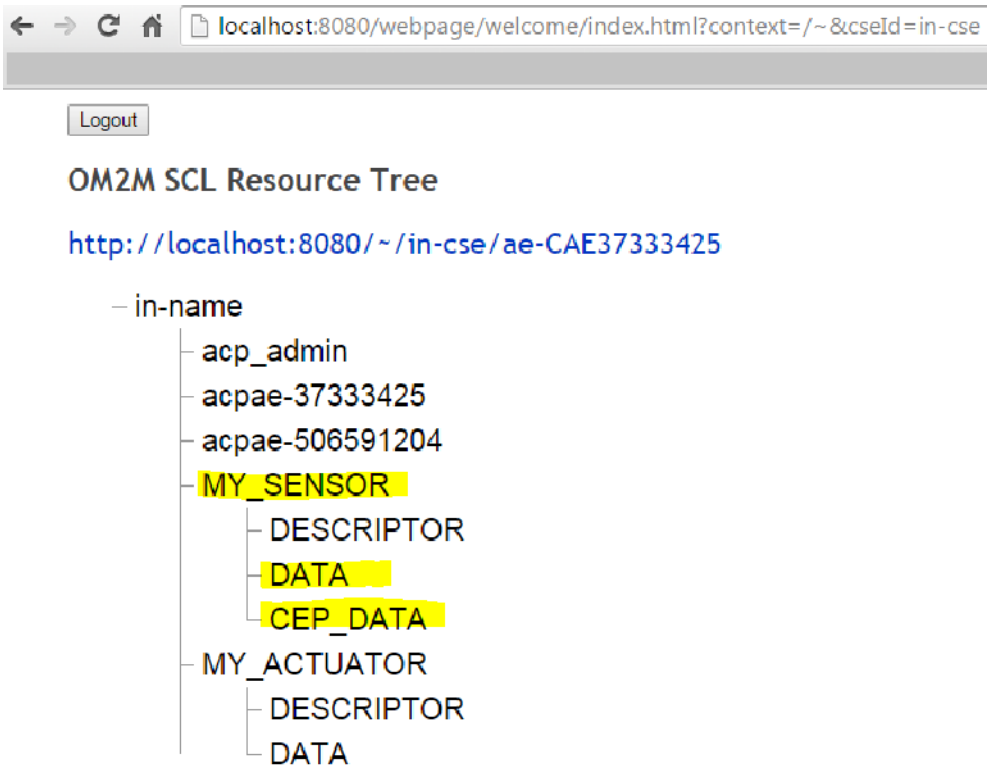
Pravila se lahko dodajajo tudi programsko v samem vtičniku platforme (Slika 4.33). To storimo tako, da jih dodamo pred samim zagonom CEP strežnika.

```
cepServer.addCepRule(sensorId, "CEP_DATA", "select * from Data.win:length(2) having avg(value) > 50");
cepServer.run();
```

Slika 4.33: Kreiranje pravila CEP (programsko).

Odpremo nov zavihek v spletnem brskalniku, vpišemo url naslov »http://localhost:8080/webpage« in se avtenticiramo (privzete vrednosti so admin:admin). Kot lahko opazimo, ima platforma OM2M povezano napravo »`MY_SENSOR`«, v kateri ima dva podatkovna modela »`DATA`« in »`CEP_DATA`«. Slednjega smo naredili mi z ustvarjanjem CEP pravila. V model »`DATA`« se

zapisujejo vsi prejeti podatki iz naprave »MY_SENSOR«, v model »CEP_DATA« pa se zapišejo podatki iz iste naprave, vendar le tisti, ki se ujamejo v prej definirano CEP pravilo (Slika 4.34).



Slika 4.34: Podatkovni modeli kreiranega vtičnika platforme OM2M.

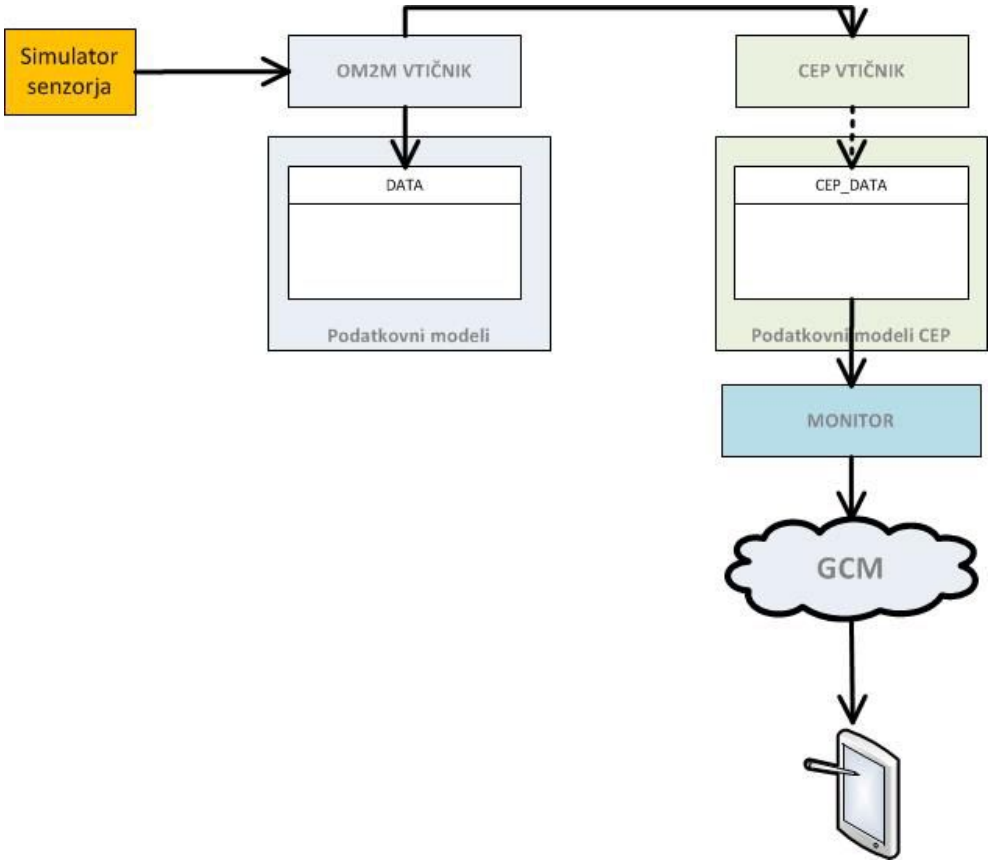
Poglavje 5

Praktičen prikaz delovanja

Možnost procesiranja kompleksnih dogodkov se v raznih IoT scenarijih izkaže kot zelo pomembna. Z nadgradnjo smo omogočili, da se namesto standardnega naročanja na podatke iz platforme, naprave naročijo samo na podatke, ki jih zanimajo in so o tem obveščene v kolikor se ti dogodki dejansko zgodijo.

Kot klasičen primer lahko vzamemo napravo za merjenje pritiska, ki je povezana s platformo OM2M. Naprava na določeni interval meri pritisk in pošiljala meritve platformi. V osnovi bi se lahko naročili na prejemanje teh podatkov, kar pa pomeni, da bi bili obveščeni vsakič, ko bi se izvedla meritev in se poslala na platformo. Z nadgradnjo pa smo omogočili, da se naročimo samo na dogodke, ki jih prejmemo v kolikor krvni tlak preseže neko mejno vrednost in na njej ostane dlje časa, medtem ko mi v resnici mirujemo.

Zgornji primer smo vzeli pod drobnogled in ga implementirali. Izdelali smo vtičnik za platformo OM2M, ki uporablja našo knjižnico za procesiranje kompleksnih dogodkov. Na vtičnik je povezan senzor, ki pošilja podatke o uporabnikovem pritisku in podatke o gibanju osebe. Te podatke vtičnik posreduje knjižnici CEP, ki pa glede na postavljeno pravilo posreduje obvestilo njemu namenjen podatkovni model. Na ta podatkovni model smo naročili aplikacijo, ki v primeru obvestila posreduje podatke na storitev GCM (Google Cloud Messaging) in s tem obvesti uporabnika na zvišan krvni tlak preko mobilne aplikacije (Slika 5.1).



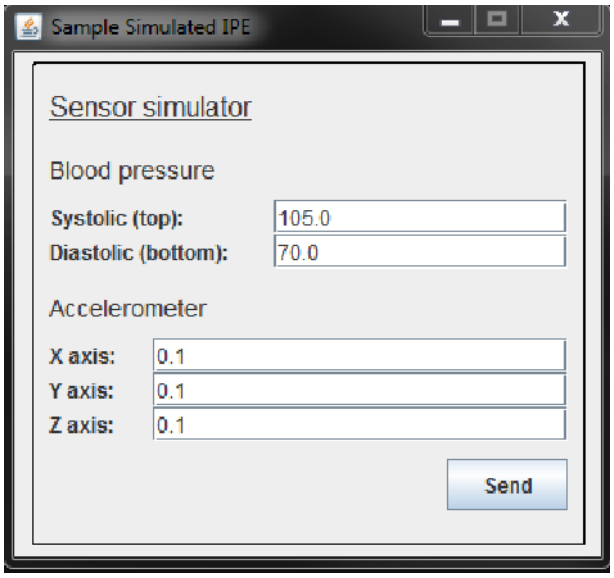
Slika 5.1: Shema praktičnega prikaza delovanja.

5.1 Simulator senzorja pritiska in pospeškometra

Simulator je bil izdelan kot grafični uporabniški vmesnik naprave, ki vsebuje pospeškometer in merilnik krvnega tlaka (Slika 5.2). Omogoča vpis in pošiljanje vrednosti meritev na platformo OM2M:

- Systolic (top) - sistolična (zgornja) vrednost meritve krvnega tlaka
- Diastolic (bottom) - diastolična (spodnja) vrednost meritve krvnega tlaka
- X axis - meritev pospeškometra po X osi

- Y axis - meritev pospeškometra po Y osi
- Z axis - meritev pospeškometra po Z osi



Slika 5.2: Simulator senzorja.

5.2 Pravilo CEP

Na platformi OM2M je bilo potrebno kreirati pravilo CEP, ki bo zaznalo, da neko zaporedje prejetih podatkov iz simulatorja pomeni visok krvni tlak in mirovanje osebe dlje časa. Tako je bilo kreirano pravilo:

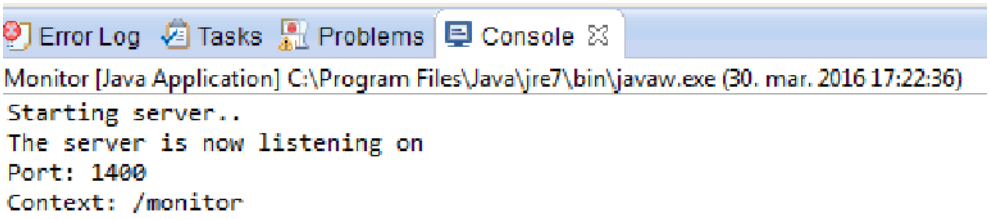
```
select avg(systolic) AS systolic, avg(diastolic) AS diastolic,
avg(x) AS x, avg(y) AS y, avg(z) AS z
from Sensor.win:length(4)
having (avg(systolic) > 140 AND avg(diastolic) > 90) AND
(avg(x) < 0.5 AND avg(y) < 0.5 AND avg(z) < 0.5 )
```

Zgoraj zapisano pravilo pomeni, da se vrednosti senzorja ujamejo v pogoj, če so zadnje štiri sistolične vrednosti v povprečju večje od 140, zadnje štiri diastolične vrednosti v povprečju večje od 90 ter da so X, Y in Z vrednosti pospeškometra v zadnjih štirih meritvah v povprečju manjše od 0.5. Torej se dogodek ujame v pravilo, če se oseba ne premika in ima v povprečju krvni pritisk, ki ustreza pretirano povišanemu krvnemu tlaku (arterijska hipertenzija) [49].

Zaradi lažjega prikaza delovanja smo v pravilu uporabili premikajoče okno, ki upošteva štiri zadnje prejete meritve in ne časovnega okna s katerim bi lahko upoštevali meritve, ki bi bile prejete v zadnjih nekaj časovnih enot.

5.3 Aplikacija Monitor

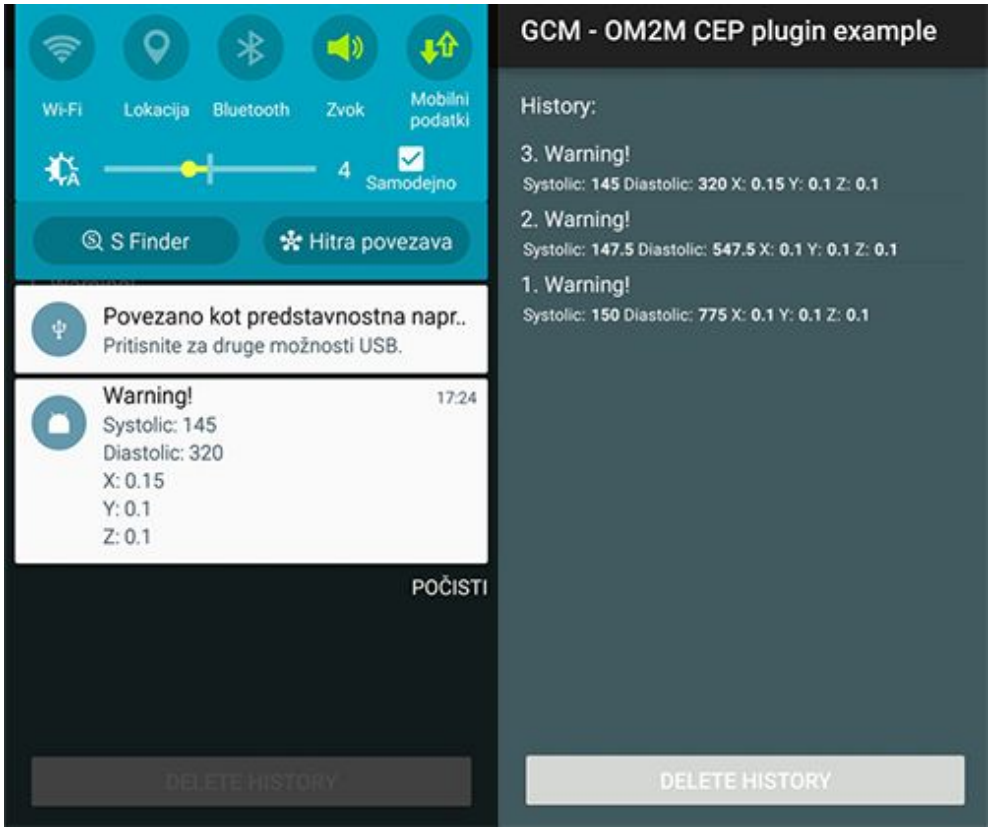
Za posredovanje podatkov uporabniku preko storitve GCM (Google Cloud Messaging) smo potrebovali aplikacijo, ki smo jo naročili na prejemanje podatov, ki ustrezajo postavljenemu pravilu. Ko aplikacija prejme obvestilo, preko storitve GCM pošlje obvestilo na mobilni telefon osebe, ki nosi naš (simulacijski) senzor (Slika 5.3).



Slika 5.3: Aplikacija Monitor.

5.4 Aplikacija za platformo Android

Za prikazovanje opozoril iz storitve GCM je potrebno na pametni napravi imeti aplikacijo, ki prejme in prikaže opozorilo uporabniku. Izdelan je bil tak primer aplikacije, ki uporabniku prikaže in shrani obvestilo o povprečnih vrednostih zadnjih štiri meritev, poslanih iz simulatorja (Slika 5.4).

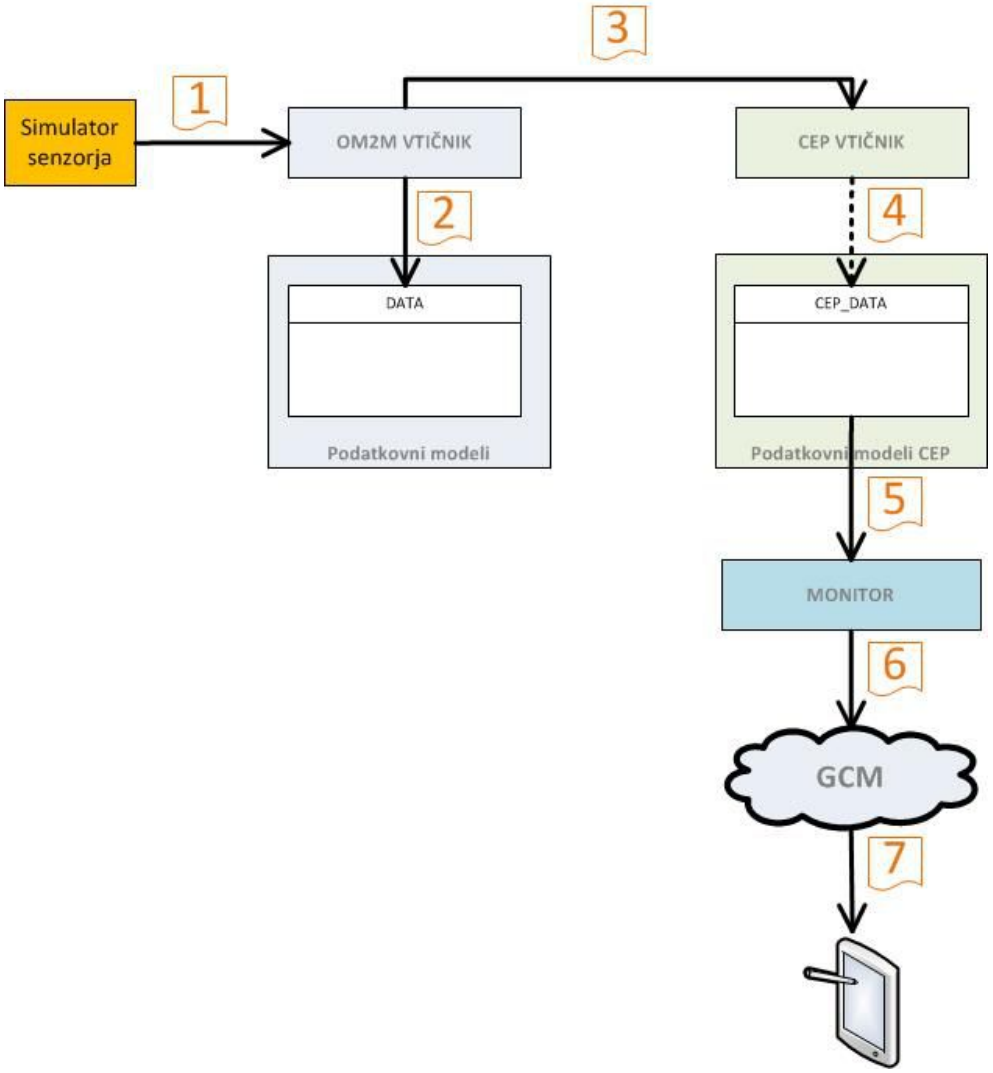


Slika 5.4: Aplikacija za platformo Android.

5.5 Struktura sporočil

Kot že rečeno je struktura sporočil znotraj OM2M platforme določena s standardom oBIX (Open Building Information Xchange). Prikazali bomo strukturo sporočil, ki se pošiljajo v našem primeru uporabe (Slika 5.5). Prikazana

so sporočila v primeru, da simulator pošlje sporočilo s sistolično vrednostjo 150.00, diastolično vrednostjo 100.00, X 0.1, Y 0.1 ter Z 0.1, pri čemer predpostavljamo, da sta povprečna sistolična in diastolična vrednost zadnjih štirih sporočil večja od 140 in 90 ter da so povprečne vrednosti X, Y in Z zadnjih štirih sporočil manjše od 0.5. To je namreč pogoj, da se bo sporočilo, ki bo uporabljeno za prikaz, ujelo v postavljeno CEP pravilo.



Slika 5.5: Struktura sporočil znotraj platforme OM2M.

5.5.1 SENZOR - OM2M VTIČNIK (1)

```
-
X-M2M-Origin: admin:admin
Content-Type: application/xml;ty=4
-
<om2m:cin xmlns:om2m="http://www.onem2m.org/xml/protocols">
    <cnf>application/obix</cnf>
    <con>
      <obj>
        <str val="150.0" name="systolic"/>
        <str val="100.0" name="diastolic"/>
        <str val="0.1" name="x"/>
        <str val="0.1" name="y"/>
        <str val="0.1" name="z"/>
      </obj>
    </con>
</om2m:cin>
```

5.5.2 OM2M VTIČNIK - DATA (2)

```
-
X-M2M-Origin: admin:admin
Content-Type: application/xml;ty=4
-
<om2m:cin xmlns:om2m="http://www.onem2m.org/xml/protocols">
    <cnf>application/obix</cnf>
    <con>
      <obj>
        <str val="150.0" name="systolic"/>
        <str val="100.0" name="diastolic"/>
        <str val="0.1" name="x"/>
```

```
        <str val="0.1" name="y"/>
        <str val="0.1" name="z"/>
      </obj>
    </con>
</om2m:cin>
```

5.5.3 OM2M VTIČNIK - CEP VTIČNIK (3)

Podatkovni objekt Sensor, ki implementira DataInterface

```
{
  double systolic = 150.0;
  double diastolic = 100.0;
  double x = 0.1;
  double y = 0.1;
  double z = 0.1;
}
```

5.5.4 CEP VTIČNIK - CEP DATA (4)

```
-
X-M2M-Origin: admin:admin
Content-Type: application/xml;ty=4
-
<om2m:cin xmlns:om2m="http://www.onem2m.org/xml/protocols">
    <cnf>application/obix</cnf>
    <con>
      <obj>
        <str name="data" val="{
         systolic=avg(systolic) zadnjih 4 meritev,
         diastolic=avg(diastolic) zadnjih 4 meritev,
         x=avg(x) zadnjih 4 meritev,
```

```
          y=avg(y) zadnjih 4 meritev,
          z=avg(z) zadnjih 4 meritev
         }" />
       </obj>
    </con>
</om2m:cin>
```

5.5.5 CEP DATA - MONITOR (5)

```
-
Accept-encoding: gzip,deflate
X-m2m-origin: /in-cse
Connection: Keep-Alive
Host: localhost:1400
User-agent: Apache-HttpClient/4.3.6 (java 1.5)
Content-type: application/xml
Content-length: 763
-
<m2m:sgn xmlns:m2m="http://www.onem2m.org/xml/protocols">
   <nev>
      <rep rn="cin_692559095">
         <ty>4</ty>
         <ri>cin-692559095</ri>
         <pi>/in-cse/cnt-60584884</pi>
         <ct>20160512T094523</ct>
         <lt>20160512T094523</lt>
         <st>0</st>
         <cnf>application/obix</cnf>
         <cs>117</cs>
         <con>
        <obj>
```

```
         <str name="data" val="{
                     systolic=avg(systolic) zadnjih 4 meritev,
                     diastolic=avg(diastolic) zadnjih 4 meritev,
                     x=avg(x) zadnjih 4 meritev,
                     y=avg(y) zadnjih 4 meritev,
                     z=avg(z) zadnjih 4 meritev
           }"/>
       </obj>
      </con>
      </rep>
      <rss>1</rss>
   </nev>
   <sud>false</sud>
   <sur>/in-cse/in-name/SENSOR/CEP_DATA/SUB_MY_SENSOR</sur>
</m2m:sgn>
```

5.5.6 MONITOR - GCM (6)

```
-
Authorization: key=AI**...**IQ
Content-Type: application/json
-
{
  "to" : "/topics/om2m",
  "data" : {
   "title" : "Obvestilo",
    "message": "{
     systolic=avg(systolic) zadnjih 4 meritev,
        diastolic=avg(diastolic) zadnjih 4 meritev,
        x=avg(x) zadnjih 4 meritev,
        y=avg(y) zadnjih 4 meritev,
```

```
        z=avg(z) zadnjih 4 meritev
    }"
  },
}
```

5.5.7 GCM - ANDROID (7)

```
-
Content-Type: application/json
-
{
   "data" : {
     "title" : "Obvestilo",
     "message": "{
        systolic=avg(systolic) zadnjih 4 meritev,
        diastolic=avg(diastolic) zadnjih 4 meritev,
        x=avg(x) zadnjih 4 meritev,
        y=avg(y) zadnjih 4 meritev,
        z=avg(z) zadnjih 4 meritev
     }"
   },
}
```

5.6 Evalvacija sistema

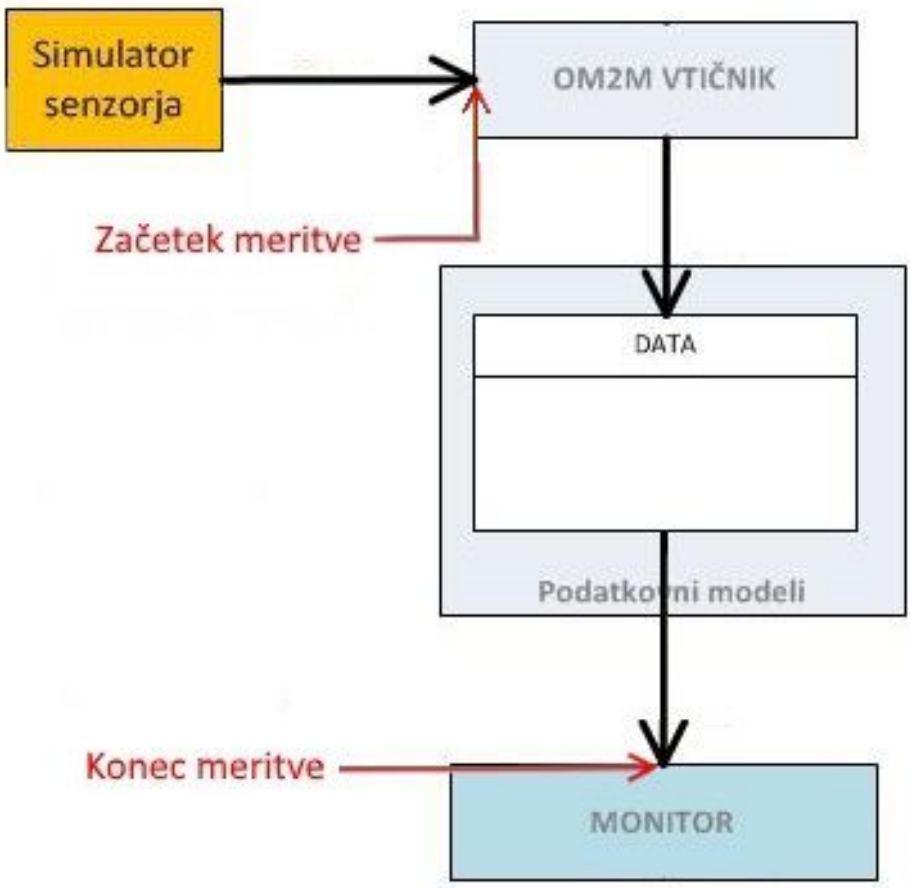
Za evalvacijo sistema smo uporabili implementirani primer uporabe, opisan v prejšnjem poglavju. Zanima nas, kako hitro platforma procesira podatke, jih posreduje naprej na naročeno napravo ter kako se ta hitrost spreminja s povečevanjem števila postavljenih CEP pravil, števila povezanih naprav na platformo in časovnega intervala pošiljanja podatkov na platformo.

Teste in meritve smo opravljali na sistemu z naslednjimi specifikacijami:

```
Windows 7 Professional (x64)
Gigabyte GA-970A-UD3P ATX AM3+
AMD Phenom™ II X6 1055T Processor 2.80 GHz
8,00 GB DDR3
AMD Radeon R9 270X
```

5.6.1 Evalvacija platforme OM2M brez knjižnice CEP

Primeru uporabe, opisanem v prejšnjem poglavju, smo dodali števce s pomočjo katerih smo izračunali, koliko časa platforma povprečno porabi za procesiranje podatka od prejema do posredovanja naprej na naročeno napravo (Monitor), kot prikazuje slika 5.6.



Slika 5.6: Evalvacija platforme OM2M brez knjižnice CEP

Meritve senzorja smo na platformo pošiljali z določenimi intervali. Pričeli smo z izračunom povprečnega časa procesiranja meritve senzorja v primeru pošiljanja podatkov na platformo z intervalom 100 ms. Na vsakih 50 poslanih podatkov iz senzorja smo naredili krajšo prekinitev in zmanjšali interval pošiljanja za 2 ms. Tako smo nadaljevali z računanjem povprečnega časa procesiranja do intervala pošiljanja 0 ms, kar pomeni, da smo vzporedno poslali petdeset podatkov iz senzorja na platformo OM2M. To smo storili za scenarije, kjer je bilo na platformo povezanih 1, 20 in 30 senzorjev.

Tabela 5.1 prikazuje izmerjene rezultate.

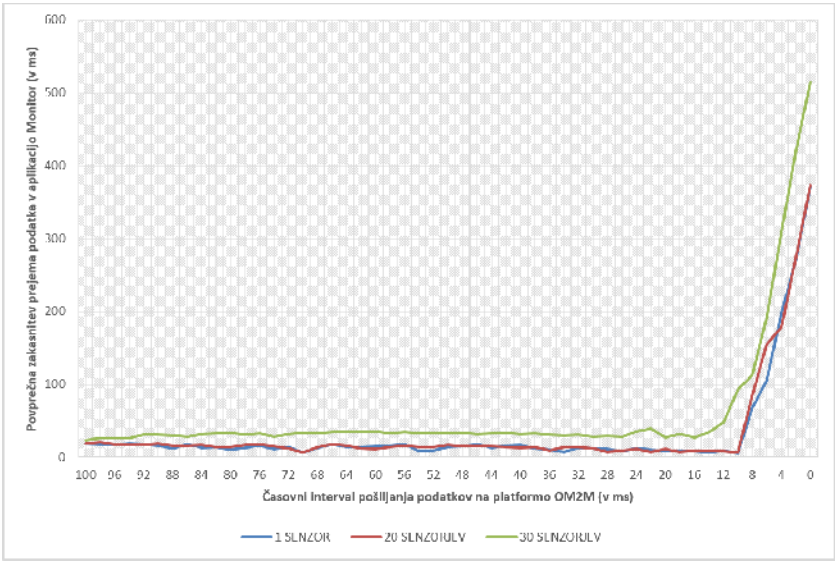
INTERVAL POŠILJANJA	1 SENZOR	20 SENZORJEV	30 SENZORJEV
100	19,5	19,54	23,8
98	18,4	21,14	26,5
96	17,06	18,36	26,56
94	18,68	17,14	26,56
92	18,42	17,18	31,54
90	15,98	18,66	31,58
88	12,72	16,08	30,2
86	17,88	15,88	28,12
84	13,88	17,38	32,56
82	14,8	14,58	33,3
80	10,92	14,46	33,7
78	13,06	16,88	31,36
76	16,3	18,3	33,1
74	11,74	15,64	28,54
72	14,62	12,9	32,18
70	6,76	6,88	34,2
68	13,7	14,54	33,54
66	18,44	17,84	35,48
64	14,04	16,34	35,4
62	14,1	12,42	35,26
60	15,24	11,84	36,12
58	16,08	14,6	33,5
56	17,74	16,12	34,92
54	8,56	14,64	33,08
52	9,88	14,38	33,86
50	14,68	17	33,54
48	15,66	15,38	34,3
46	18,5	15,58	32,66
44	13,66	15,44	33,38
42	16,7	14,46	33,78
40	17,58	13,82	32,38
38	12,92	14,88	33,2
36	11,04	9,84	31,2
34	8,18	14,62	30,76
32	13,26	14,52	31,52
30	12,76	12,28	28,6
28	11,94	8,16	29,38
26	8,76	9,62	28,92
24	13,02	12,04	36,08
22	10,48	7,42	39,96
20	9,24	12	27,76
18	10,08	7,5	32,48
16	8,44	10	27,56
14	7,26	9,26	35,08
12	10,08	8,48	48,18
10	5,94	7	93,48
8	67,84	84,86	112,34
6	105,6	155,2	191,24
4	199,08	179,02	310,12
2	271,94	274,44	420,78
0	371,22	373,54	514,96
POVPREČJE	32,04627451	33,53098039	60,75686275

Tabela 5.1: Evalvacija sistema brez knjižnice CEP

Ugotovili smo, da je zakasnitev podatka pri enem ali dvajsetih povezanih senzorjev približno enaka. Pri tridesetih povezanih senzorjih je zakasnitev podatka v povprečju večja za približno 27,23 ms od dvajsetih povezanih senzorjev. Pri intervalih pošiljanja od 100 do 12 ms, se zakasnitev ne razlikuje veliko, pri intervalih manjših od 12 ms pa zakasnitev poskoči. Pri vzporednem pošiljanju podatkov na platformo (interval 0 ms) je zakasnitev v povprečju 30,5 krat večja kot pri pošiljanju z 12 ms intervalom, ne glede na število povezanih senzorjih.

Minimalna zakasnitev je bila 5,94 ms pri intervalu pošiljanja 10 ms, maksimalna zakasnitev 514,96 ms pri intervalu pošiljanja 0 ms, povprečje vseh meritev pa 42,11 ms.

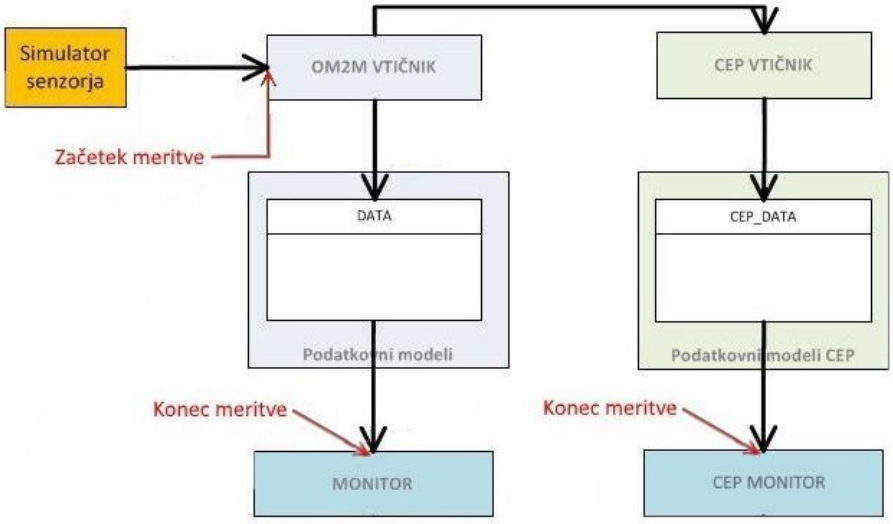
Slika 5.7 prikazuje povprečno zakasnitev prejema podatka v aplikacijo, ki je naročena na podatkovni model (Monitor), glede na časovni interval pošiljanja podatkov na platformo OM2M pri enem, dvajset in tridesetih povezanih senzorjih.



Slika 5.7: Povprečna zakasnitev prejema podatka v aplikacijo, ki je naročena na podatkovni model glede na časovni interval pošiljanja podatkov na platformo OM2M pri enem, dvajsetimi in tridesetimi povezanimi senzorji

5.6.2 Evalvacija platforme OM2M s knjižnico CEP

Za evalvacijo platforme s knjižnico CEP smo prav tako uporabili prej opisani primer uporabe. Implementirali smo števce, s katerimi smo izmerili čas, ki ga platforma OM2M potrebuje za procesiranje podatka od prejema do posredovanja naprej na naročeno napravo (v nadaljevanju aplikacija »`Monitor`«) ter čas, ki ga potrebuje za procesiranje ter ujemanje s pravilom CEP od prejema podatka do posredovanja le tega naprej na naročeno napravo (v nadaljevanju aplikacija »`CEP Monitor`«) (Slika 5.8).



Slika 5.8: Evalvacija platforme OM2M s knjižnico CEP

Prav tako smo meritve senzorja na platformo pošiljali v intervalih od 100 ms do 0 ms, ki smo ga zmanjševali za korak 2ms. Za izračun povprečne zakasnitve smo uporabili 50 meritev za vsak interval. Meritve smo izvajali za scenarij, kjer je bil na platformo povezan eden ali petdeset senzorjev ter eno, dvajset in petdeset postavljenih pravil CEP. Za bolj realno oceno zakasnitve sta bila senzor in pravilo CEP, nad katerim so se izvajale meritve v platformo, dodana zadnja.

Tabela 5.2 prikazuje izmerjene rezultate z enim definiranim pravilom.

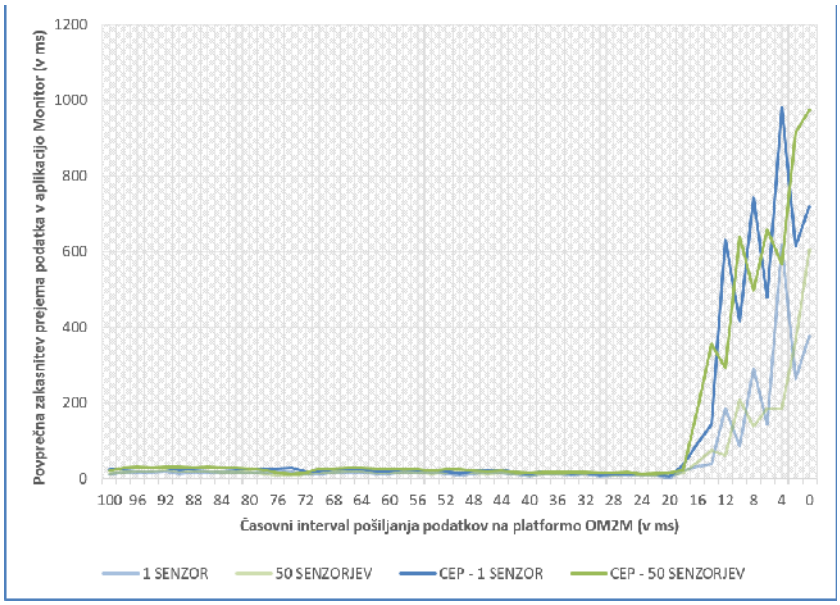
INTERVAL POŠILJANJA	1 SENZOR	50 SENZORJEV	CEP - 1 SENZOR	CEP - 50 SENZORJEV
100	12,68	14,34	24,38	22,24
98	17,76	18,18	26,64	28,5
96	18,16	19,3	29,62	30,22
94	18,54	19,68	28,42	29,88
92	19,42	20,7	29,62	31,94
90	13,4	19,18	23,22	30,88
88	18,26	19,36	27,78	29,92
86	18,28	19,82	28,52	31,56
84	18,1	17,3	28,52	28,28
82	17,62	18,18	26,64	28,28
80	14,8	17,4	25,22	27,2
78	18,04	14,18	27,9	23,84
76	17,62	7,44	27,7	14,88
74	18,08	8,38	28,4	12,32
72	12,02	11,92	19,84	16,66
70	12,36	18,24	19,42	26,78
68	17,3	18,38	25,16	27,22
66	17,72	19,04	25,56	28,32
64	18,46	18,86	26,16	28,42
62	14,74	18,04	19,24	27,48
60	14,52	18,44	20,34	27,84
58	17,46	18,12	24,36	25,84
56	17,64	17,72	24,16	26,06
54	17,04	14,32	23,74	20,84
52	14,42	17	20,42	25,6
50	8,06	17,32	13,12	24,82
48	14,42	15,12	20,9	20,62
46	16,74	14,2	24,28	19,46
44	15,38	15,12	22,68	21,06
42	11,72	12,94	17,22	18,3
40	5,92	10,92	11,04	15,16
38	13,28	11,94	18,72	18,38
36	12,54	13	18,2	17,86
34	9,82	13	13,72	18,4
32	11,26	12,06	15,98	17,22
30	6,46	11,48	10,26	16,06
28	8,52	10,16	11,76	14,8
26	10,06	12,8	14,06	18,46
24	8,06	9,08	11,86	12,74
22	8,9	10,48	12,34	14,54
20	2,8	9,5	6,22	14,84
18	20,9	14,92	37,5	22,46
16	33,82	43,58	94,04	185,54
14	39,06	75,72	144	355,54
12	185,56	61,04	631	293,06
10	87,44	209,12	417,22	639,08
8	290,28	138,9	742,94	498,18
6	144,62	186,6	478,72	658,78
4	617,88	185,52	980,8	566,06
2	264,3	362,92	613,74	914,7
0	376,78	604,74	720,04	974
POVPREČJE	51,7454902	49,13137255	112,4184314	118,4533333

Tabela 5.2: Evalvacija sistema s knjižnico CEP z enim definiranim pravilom CEP

Ugotovili smo, da število senzorjev ne vpliva veliko na časovno zakasnitev podatka v obeh aplikacijah pri enem definiranem CEP pravilu. V aplikacijo Monitor je v povprečju podatek prišel z 50,4 ms zamudo, v aplikacijo CEP Monitor pa z 115,4 ms zamudo. Pri intervalih pošiljanja od 100 do 20 ms se zakasnitve ne razlikujejo veliko, pri intervalih pošiljanja manjših od 20 ms pa se zakasnitev v obe aplikaciji poveča.

Minimalna zakasnitev ne glede na število povezanih senzorjev je bila 2,8 ms pri intervalu pošiljanja 20ms, maksimalna zakasnitev 980,8 ms pri intervalu pošiljanja 4 ms, povprečje vseh zakasnitev pa je bila 82,9 ms.

Slika 5.9 prikazuje povprečno zakasnitev prejema podatka v aplikacijo Monitor ter CEP Monitor glede na časovni interval pošiljanja podatkov na platformo OM2M pri enem uporabljenem CEP pravilu za 1 in 50 povezanih senzorjev.



Slika 5.9: Povprečna zakasnitev prejema podatka v aplikacijo Monitor ter CEP Monitor glede na časovni interval pošiljanja podatkov na platformo OM2M pri enem uporabljenem CEP pravilu za 1 in 50 povezanimi senzorji

Tabela 5.3 prikazuje izmerjene rezultate z dvajsetimi definiranimi pravili.

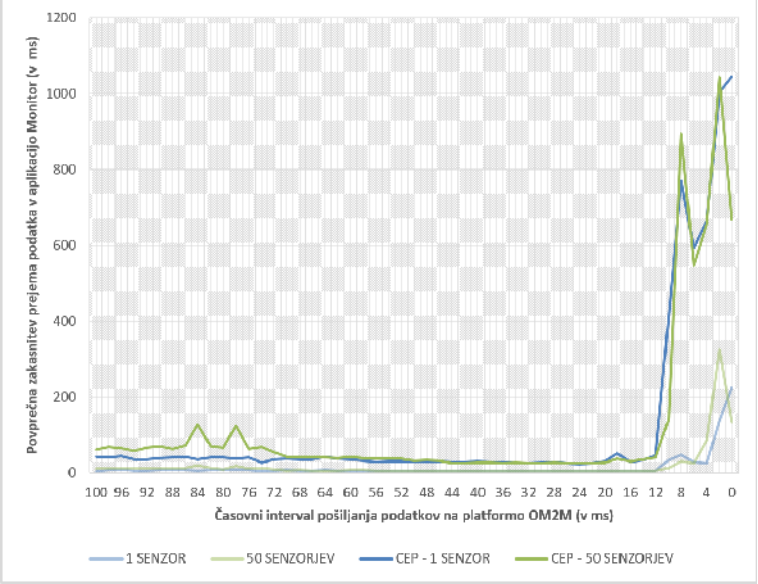
INTERVAL POŠILJANJA	1 SENZOR	50 SENZORJEV	CEP - 1 SENZOR	CEP - 50 SENZORJEV
100	3,7	11,3	42,34	61,58
98	7,64	12,28	40,72	67,44
96	8,38	11,54	43,04	64,1
94	6,02	10,02	35,02	57,78
92	6,2	12,74	35,64	66,64
90	6,9	13,02	39,26	69,52
88	6,94	10,74	39,6	62,18
86	6,84	12,3	42,04	70,96
84	6,26	19,44	35,88	127,98
82	6,48	11,88	39,86	70,46
80	6,68	8,82	39,9	66,02
78	6,46	17,3	37,32	122,88
76	6,52	10,14	39,44	62,72
74	4,44	11,68	26,24	68,3
72	5,54	9,62	35,14	54,42
70	6,56	6,04	37,52	42,36
68	5,58	6,44	34,9	40,78
66	5,92	6,28	34,76	40,8
64	6,4	6,24	41,28	39,58
62	5,8	5,9	37,52	37,86
60	5,5	6,9	35,36	41,88
58	4,84	6,44	31,8	38,88
56	4,72	5,66	28,52	37,3
54	4,78	5,48	30,54	37
52	4,7	5,8	29,92	37,06
50	4,16	4,8	27,64	31,34
48	5,4	4,66	29,84	33,72
46	4,9	5,26	30,06	32,36
44	4,5	4,28	27,62	24,92
42	4,34	3,64	27,3	24,66
40	4,38	4,86	29,1	26,92
38	4,4	3,9	27,56	25,62
36	4	4,04	27,88	25,02
34	4,18	3,94	25,3	26,04
32	3,96	3,98	24,92	25,2
30	4,02	3,9	25,66	24,46
28	4,28	3,88	27,48	25,66
26	3,58	3,5	24,9	25,34
24	3,16	3,38	22,94	24,72
22	3,1	3,48	23,96	24,88
20	3,64	3,62	29,34	27,04
18	5,98	4,58	51,02	35,98
16	3,64	3,74	27,3	29,04
14	4,42	4,44	33,9	35,72
12	4,9	5,2	44,88	40,62
10	32,66	12,06	411,64	141,56
8	46,64	30,34	770,72	894,46
6	27,38	25,12	593,14	546,36
4	25,34	85,1	664,9	656,42
2	136,84	325,06	1002,68	1043,86
0	223,6	134,74	1044,88	668,4
POVPREČJE	14,2588235	18,42156863	117,4533333	117,7803922

Tabela 5.3: Evalvacija sistema s knjižnico CEP z dvajsetimi definiranimi pravili CEP

Pri dvajsetih definiranih CEP pravilih smo opazili, da so se povečale razlike v zakasnitvah med aplikacijo Monitor in CEP Monitor. Pri enem povezanem senzorju je ta razlika v povprečju 103,2 ms, pri petdesetih povezanih senzorjih pa 99,36 ms. Tudi pri dvajsetih definiranih CEP pravilih smo opazili, da število senzorjev ne vpliva veliko na časovno zakasnitev podatka v obeh aplikacijah. Zakasnitve se ne razlikujejo veliko pri intervalih pošiljanja od 100 do 12 ms, pri intervalih pošiljanja manjših od 12 ms pa zakasnitve tudi v tem scenariju poskočijo.

Minimalna zakasnitev ne glede na število povezanih senzorjev je bila 3,1 ms pri intervalu pošiljanja 22 ms, maksimalna zakasnitev 1044,88 ms pri intervalu pošiljanja 0 ms, povprečje vseh zakasnitev pa je bilo 67 ms.

Slika 5.10 prikazuje povprečno zakasnitev prejema podatka v aplikacijo Monitor ter CEP Monitor glede na časovni interval pošiljanja podatkov na platformo OM2M pri dvajsetih uporabljenih CEP pravilih za 1 in 50 povezanih senzorjev.



Slika 5.10: Povprečna zakasnitev prejema podatka v aplikacijo Monitor ter CEP Monitor glede na časovni interval pošiljanja podatkov na platformo OM2M pri dvajsetih uporabljenih CEP pravilih za 1 in 50 povezanih senzorjev

Tabela 5.4 prikazuje izmerjene rezultate s petdesetimi definiranimi pravili.

INTERVAL POŠILJANJA	1 SENZOR	50 SENZORJEV	CEP - 1 SENZOR	CEP - 50 SENZORJEV
100	30,34	70	521,88	1797,94
98	50,14	71,68	1261,94	1828,64
96	43,78	56,6	1150,26	1525
94	52,26	55,5	1322,76	1606,62
92	53,12	53,8	1364,8	1471,82
90	39,58	54,94	1149,76	1487,92
88	60,28	58,86	1752,14	1661,8
86	61,58	57,04	1831	1618,44
84	83,36	65,22	2493,06	1779
82	70,32	64,66	2148	1886,88
80	89,54	67,4	2379,26	1947,48
78	313,92	185,7	6604,44	8339,6
76	219,82	2430,56	10256,7	15961,44
74	224,86	1044,24	12849,06	17699,04
72	229,16	171,68	14162,56	18812,76
70	258,52	219,82	16792,34	19892,78
68	240,02	207,22	18054,06	22630,14
66	228,62	214,24	18616,18	27226,14
64	231,98	239,3	19341,08	36464,68
62	266,12	254,78	21885,22	57702,24
60	263,3	352,46	25000,96	93457,36
58	266,06	451,26	32815,86	130351,92
56	297,14	751,44	46221,34	168180,3
54	251,92	991,58	68390,04	226024,1
52	268,06	1175,12	104992,52	283188,8
50	498,76	1762,46	136305,94	333881,14
48	929,44	4358,96	175259,42	387719,26
46	1117,32	6755,26	223290,58	445848,98
44	2209,66	12842,44	271917,58	516371,36
42	2265,74	17997,86	331251,04	608502,36
40	2240,72	21357,22	396087,52	692042,28
38	4666,4	19918,72	470983,96	767219,3
36	7743,02	24630,04	531660,44	857488,7
34	14521,28	31359,26	606819	940284,94
32	20134,06	39867,54	673060,6	997027,14
30	24312,6	39426,84	764960,52	1088495,16
28	28929,38	54564,94	840852,76	1176616,78
26	36066,7	55023,52	917459,98	1260638,2
24	52299,68	64692,44	996669,44	1334534,16
22	46271,38	78040,74	1079609,66	1412049,4
20	58118,62	87142,58	1154962,68	1460917
18	58729,28	101216,84	1209718,14	1507119,14
16	66343,98	108973,92	1271264,22	1551389,76
14	74347,46	119145,18	1325941,72	1619770,82
12	78593,92	126452,4	1373775,78	1697290,96
10	89939,14	132575,1	1417244,46	1728111,58
8	95694,18	140795,14	1449200,42	1758348,66
6	107650,88	146076,22	1465409,1	1772021,94
4	118061,66	164336,48	1488314,36	1787746,6
2	127510	169813,14	1498057,96	1792388,7
0	139857,88	187485,22	1498843,68	1797283,58
POVPREČJE	24769,5478	38548,54039	470632,9055	635993,1518

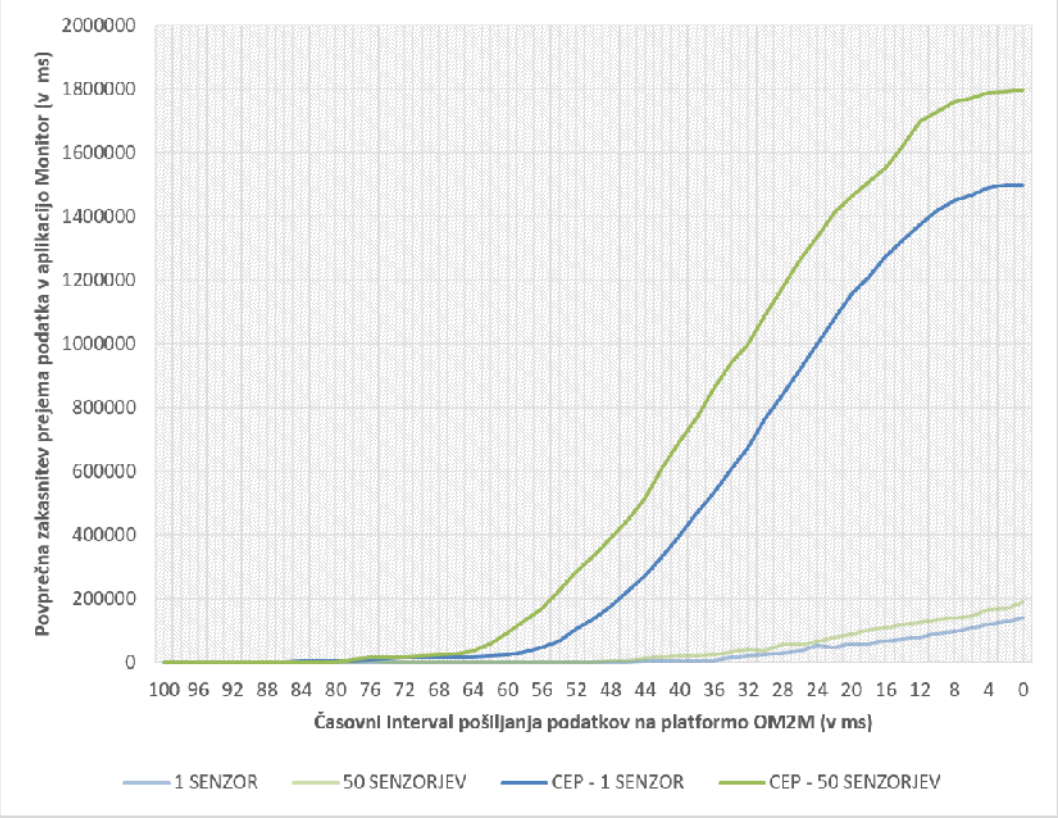
Tabela 5.4: Evalvacija sistema s knjižnico CEP s petdesetimi definiranimi pravili CEP

Pri petdesetih definiranih CEP pravilih smo ugotovili, da se vse zakasnitve zelo povečajo (Slika 5.11). Da podatek prispe v aplikacijo Monitor pri enem definiranem CEP pravilu in enem povezanem senzorju, potrebuje v najboljšem scenariju 2,8 ms, pri petdesetih definiranih CEP pravilih in enem povezanem senzorju pa 30,34 ms, kar pomeni 10,84 krat večjo zakasnitev. Za aplikacijo CEP Monitor je ta scenarij še slabši. Pri enem definiranim CEP pravilu in enem povezanem senzorju v najboljšem primeru podatek potrebuje 6,22 ms da prispe v aplikacijo CEP Monitor, pri petdesetih definiranih pravilih CEP in enem povezanem senzorju pa v najboljšem primeru potrebuje 521,88 ms, kar pomeni 83,9 krat večjo zakasnitev.

V tem scenariju smo opazili, da število senzorjev vpliva na časovno zakasnitev v aplikaciji Monitor. 10 sekundna zakasnitev se pri petdeset povezanih senzorjih preseže pri intervalu pošiljanja 44 ms, ki se pri enem povezanem senzorju preseže šele pri intervalu pošiljanja 34 ms.

Ne glede na število povezanih senzorjev se zakasnitve v aplikaciji CEP Monitor začnejo povečevati pri intervalu pošiljanja manjših od 76 ms. V aplikaciji Monitor zakasnitve pričnejo eksponentno naraščati pri intervalu pošiljanja 60 ms, medtem, ko v aplikaciji CEP Monitor že pri intervalu pošiljanja 66 ms. V obeh aplikacijah zakasnitev preneha eksponentno naraščati v intervalih pošiljanja, manjših od 12 ms.

Minimalna zakasnitev ne glede na število povezanih senzorjev je bila 30,34 ms pri intervalu pošiljanja 100 ms, maksimalna zakasnitev kar 1797283,58 ms (30 min) pri intervalu pošiljanja 0 ms, povprečje vseh zakasnitev pa je bila 292486 ms (4,9 min). Spodnji graf prikazuje povprečno zakasnitev prejema podatka v aplikacijo Monitor ter CEP Monitor glede na časovni interval pošiljanja podatkov na platformo OM2M pri petdesetih uporabljenih CEP pravilih za 1 in 50 povezanih senzorjev.



Slika 5.11: Povprečna zakasnitev prejema podatka v aplikacijo Monitor ter CEP Monitor glede na časovni interval pošiljanja podatkov na platformo OM2M pri petdesetih uporabljenih CEP pravilih za 1 in 50 povezanih senzorjev

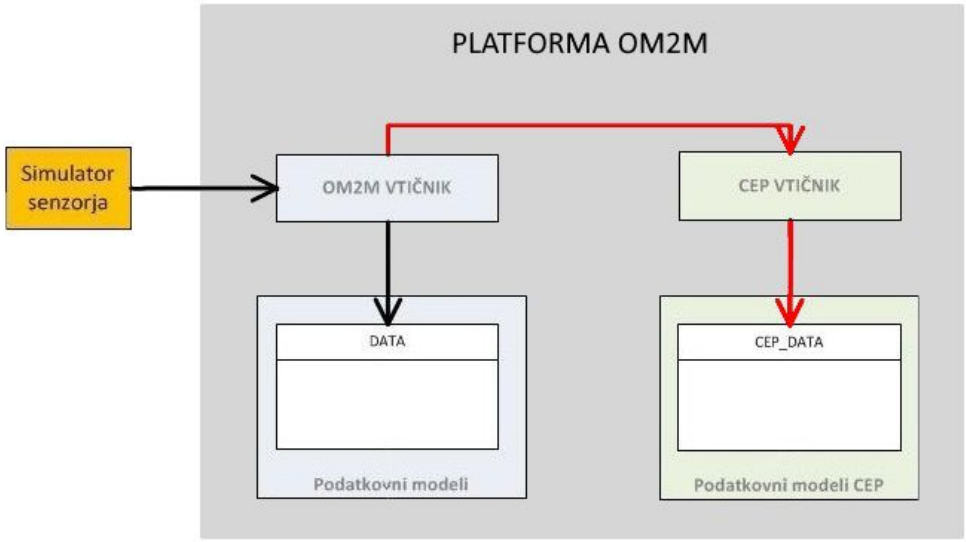
V zgornjem primeru razviti vtičnik ob vsakem prejemu podatka preveri petdeset pravil CEP, kar pomeni, da vtičnik v zadnjih šestih intervalih pošiljanja (10, 8, 6, 4, 2 in 0 ms) mora skupno preveriti 15000 pravil CEP v 1.5 sekundah, brez upoštevanja časovnega zamika med intervali. Pri takem scenariju, kjer je potrebno večje število definiranih pravil in pri tako kratkih intervalih pošiljanja podatkov na platformo, pride do nezanemarljivih zakasnitev. Problem bi lahko rešili z razdelitvijo preverjanja pravil na več enot paralelno.

Poglavje 6

Splošnost razvitega pristopa

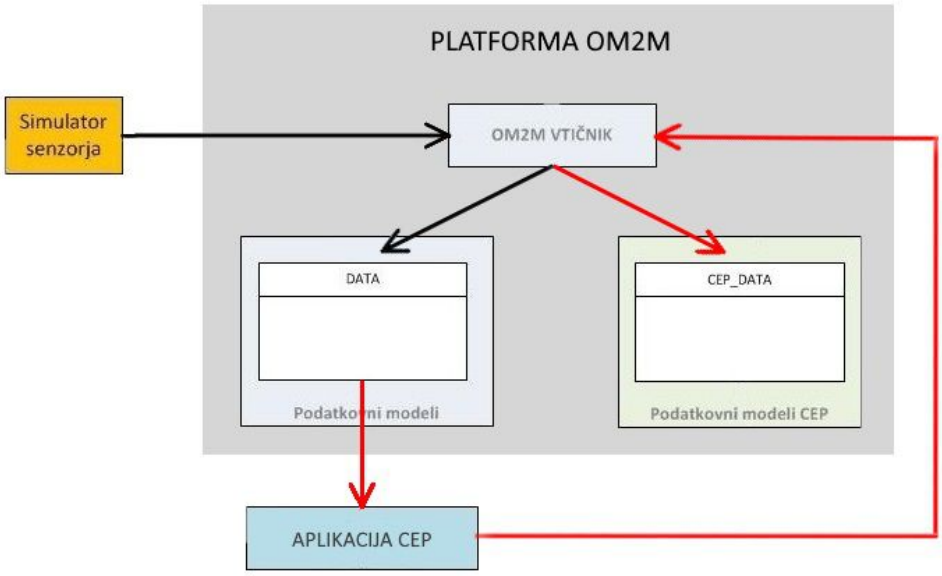
Vtičnik za procesiranje kompleksnih dogodkov je namenjen le platformi OM2M, saj uporablja njene razrede in funkcionalnost za prejemanje sporočil v vtičnik in pošiljanje naprej v določen podatkovni model. S tem ne spreminja funkcionalnosti platforme OM2M, vendar jo razširja. Tak pristop razvitega vtičnika smo uporabili, saj smo želeli pridobiti na hitrosti procesiranja in skrajšati celotno pot sporočila od prejema do zapisa v podatkovni model. Bolj splošen in bolj vsestransko uporaben v različnih platformah kot bi želeli, da je vtičnik, daljša bi morala biti pot prejetega sporočila in s tem večja zakasnitev podatka.

Razvita knjižnica oziroma vtičnik se v projekt platforme OM2M vključi med samim grajenjem in konfiguracijo, kar pomeni, da je vtičnik v času zagona del platforme (Slika 6.1). S tem omogočimo, da je podatkovna pot čim krajša.



Slika 6.1: Podatkovna pot med vtičnikom platforme OM2M in nadgradnjo CEP

Druga možnost bi bila, da bi izbrali bolj splošen pristop, ki bi nam omogočal razviti knjižnico za uporabo v različnih platformah. Za dosego teh ciljev bi morali vtičnik razviti tako, da končni izdelek ne bi bil del same platforme. Rešitev bi bila izdelati aplikacijo kot je aplikacija Monitor v praktičnem prikazu delovanja s tem, da bi procesiranje kompleksnih dogodkov prestavili v to aplikacijo in v primeru ujema dogodka v pravilo CEP to posredovali nazaj na izbrano platformo (Slika 6.2). S tem bi omogočili povezljivost aplikacije z različnimi platformami IOT, vendar pa občutno podaljšali podatkovno pot.



Slika 6.2: Podatkovna pot med vtičnikom platforme OM2M in nadgradnjo CEP v primeru bolj splošnega pristopa implementacije

6.1 Repozitorij vtičnika in vključitev v uradni repozitorij platforme OM2M

Razvoj vtičnika za procesiranje kompleksnih dogodkov smo vodili preko brezplačnega git repozitorija GitHub [50]. Repozitorij [51] vsebuje tako celoten vtičnik kot tudi iz njega zgrajeno knjižnico om2m-cep.jar. Vsebuje splošen opis vtičnika, navodila za vključitev knjižnice v projekt platforme OM2M, seznam vseh funkcij, ki jih vtičnik ponuja, reference in povezave do uporabljenih knjižnic (Esper, H2 Database Engine, Jetty) ter primer uporabe knjižnice s pomočjo grafičnega uporabniškega vmesnika [52] in direktno uporabo v vtičniku platforme OM2M [53].

Na uradnem forumu platforme OM2M smo razvijalce obvestili o izdelanem vtičniku, pozvali k uporabi in povprašali po možnosti za vključitev razširitve v uradni repozitorij platforme OM2M. Kot omenjeno, razširitev za

procesiranje kompleksnih dogodkov uporablja knjižnici H2 Database Engine ter Jetty, ki sta izdani tako kot sama platforma OM2M pod licenco EPL (Eclipse Public License) in knjižnico Esper, ki je izdana pod licenco GPL (General Public License).

Vključitev vtičnika v uradni repozitorij platforme OM2M ni mogoča, saj licenca GPL, pod katero je izdana knjižnica Esper, ni kompatibilna z licenco EPL, pod katero je izdana platforma OM2M [54].

Poglavje 7

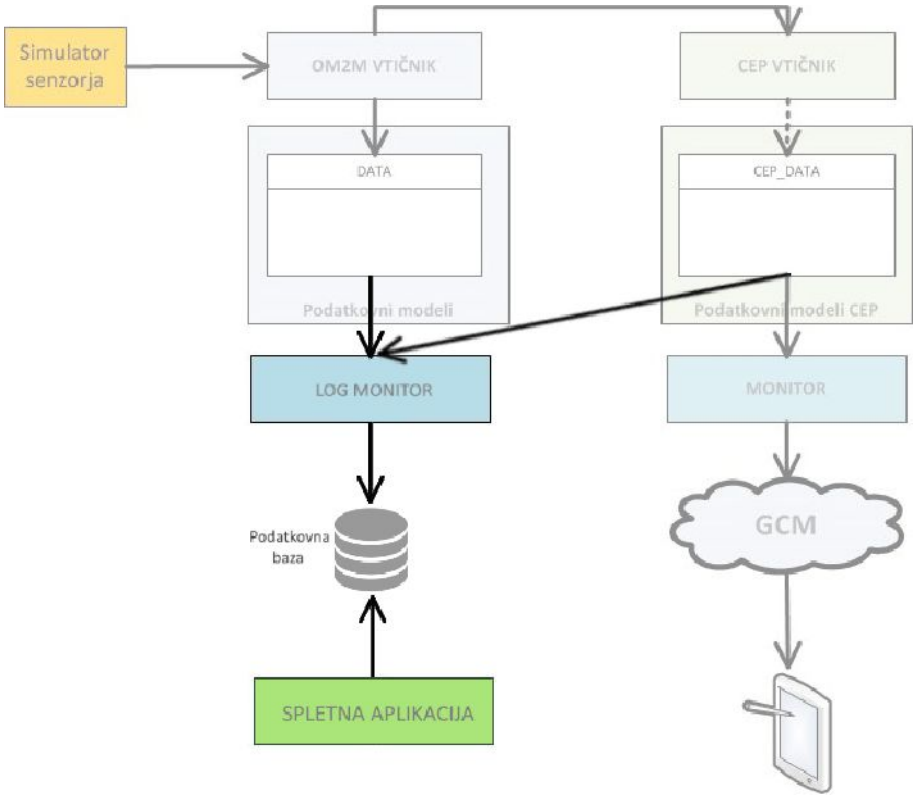
Zaključek

Razviti vtičnik za procesiranje kompleksnih pogojev v platformi OM2M izpolnjuje vse tehnične in vsebinske zahteve, ki so bile določene na začetku. Vtičnik ne posega v obstoječo platformo OM2M in se ga vključi kot samostojen vtičnik ali knjižnica, uporablja izključno odprtokodne knjižnice (Esper, Jetty in H2 Database Engine), razvijalcem omogoča lažje razhroščevanje in jim daje svobodo pri kreiranju pravil in procesiranju kompleksnih dogodkov. Omogoča dodajanje neomejenega števila CEP pravil glede na povezano napravo, urejanje le teh programsko ali preko grafičnega vmesnika ter omogoča naročanje naprav na dogodke, ujete v CEP pravilo.

Največja pomanjkljivost vtičnika je, da eno CEP pravilo lahko prejema in obdeluje podatke samo iz enega vira. To pomeni, da uporabnik ne more kreirati CEP pravila, ki bi lovil kompleksne dogodke iz več senzorjev oziroma naprav hkrati. Lovljenje takih dogodkov bi lahko implementirali tako, da več virov združimo v enega, ki podatke združuje in pošilja na platformo. Kot primer: v praktičnem prikazu delovanja implementirani simulator ne pošilja podatkov iz senzorja pritiska in pospeškometra posebej, vendar jih združi in pošlje meritve obeh senzorjev na platformo OM2M v enem sporočilu.

V praktičnem primeru uporabe smo izdelali simulator senzorja, vtičnik, aplikacijo Monitor in aplikacijo za pametne telefone z operacijskim sistemom Android. V tem poglavju smo želeli prikazati delovanje nadgradnje CEP in

podatkovni tok. Da bi bila shema popolna, bi lahko izdelali še eno aplikacijo Log monitor, ki bi se naročila na podatke v vseh podatkovnih modelih. Z njo bi v podatkovno bazo shranjevali vse prejete podatke na platformo OM2M. V nadaljevanju bi lahko izdelali spletno aplikacijo, s katero bi na uporabniku prijazen način prikazovali zgodovino prejetih podatkov na platformo (Slika 7.1).



Slika 7.1: Beleženje zgodovine v podatkovni bazi in prikaz podatkov

Pomanjkljivost, za katero smo izvedeli, ko je bil vtičnik že implementiran pa je bila nezmožnost vključitev vtičnika v uradni repozitorij platforme OM2M, saj licenca GPL, pod katero je izdana knjižnica Esper, ni kompatibilna z licenco EPL, pod katero je izdana platforma OM2M. To pomanjkljivost bi lahko odpravili v primeru uporabe druge knjižnice za procesiranje kompleksnih dogodkov s ponovnim razvojem vtičnika.

Literatura

[1] GSM Association. 2014. Understanding the Internet of Things (IoT) Dostopno na: http://www.gsma.com/connectedliving/wp-content/uploads/2014/08/cl_iot_wp_07_14.pdf

[2] Marko Bajec, Vlado Stankovski, Tomaž Vidonja. 2015. IJU 2015: Potencial interneta stvari. Dosto- pno na: http://iju2015.iju-konferenca.si/upload/Predstavitve/MRDP3/M.Bajec.pdf

[3] J. David Zook, Norbert Schroeder. 2006. Sensors as Information Transducers. Dosto- pno na: http://arxiv.org/ftp/arxiv/papers/0804/0804.0814.pdf

[4] IoT platforme. Dostopno na: https://blog.profitbricks.com/top-49-tools-internet-of-things

[5] IoT platforme. Dostopno na: http://postscapes.com/internet-of-things-platforms

[6] Odprtokodne platforme za IoT. Dostopno na: http://bbvaopen4u.com/en/actualidad/open-source-internet-things-platforms-and-applications-developers

[7] OM2M: Standardized M2M service platform. Dostopno na: http://www.eclipse.org/om2m

[8] M. Ben Alaya, Y. Banouar, T. Monteil, C. Chassot, K. Drira. 2014. OM2M: Extensible ETSI-compliant M2M Service Platform with Self-

configuration Capability. Dostopno na: http://www.sciencedirect.com/science/article/pii/S1877050914007364

[9] Michael Eckert, François Bry. 2009. Complex Event Processing (CEP). Dostopno na: http://www.researchgate.net/publication/225618561_Complex_Event_Processing_(CEP)

[10] Gianpaolo Cugola, Akessandro Margara. 2012. Processing Flows of Information: From Data Stream to Complex Event Processing. Dostopno na: http://home.deib.polimi.it/cugola/Papers/cep_survey.pdf

[11] ThingWorx. Dostopno na http://www.thingworx.com

[12] Jasper. Dostopno na https://www.jasper.com/

[13] Xively. Dostopno na https://xively.com

[14] Axiros. Dostopno na http://www.axiros.com/

[15] OpenMTC. Dostopno na http://www.open-mtc.org/

[16] AllJoyn. Dostopno na https://allseenalliance.org/framework

[17] WSO2. Dostopno na http://wso2.com/

[18] IoTivity. Dostopno na https://www.iotivity.org/

[19] Microsoft StreamInsight. Dostopno na https://msdn.microsoft.com/en-us/library/ee362541(v=sql.111).aspx

[20] Axibase Time Series Database. Dostopno na https://axibase.com/products/axibase-time-series-database/

[21] WSO2 CEP. Dostopno na http://wso2.com/products/complex-event-processor/

[22] SQLstream Blaze. Dostopno na http://www.sqlstream.com/blaze/

[23] Esper. Dostopno na http://www.espertech.com/esper/

[24] Jetty. Dostopno na http://www.eclipse.org/jetty/

[25] Primerjava Jetty z ostalimi spletnimi strežniki. Dostopno na https://en.wikipedia.org/wiki/Comparison_of_application_servers

[26] Relacijska Podatkovna baza H2 Database Engine. Dostopno na http://www.h2database.com/

[27] Primerjava H2 Database Engine z ostalimi podatkovnimi bazami. Dostopno na http://www.h2database.com/html/features.html#comparison

[28] Google Cloud Messaging. Dostopno na https://cloud.google.com/

[29] Hua-Dong Ma. 2011. Internet of Things: Objectives and Scientific Challenges. Dostopno na: http://link.springer.com/article/10.1007%2Fs11390-011-1189-5#/page-1

[30] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. Dostopno na: http://ac.els-cdn.com/S0167739X13000241/1-s2.0-S0167739X13000241-main.pdf?_tid=c26339a8-4363-11e6-8847-00000aacb35f&acdnat=1467800822_207751fba55f8fb5386a85d73d5a4259

[31] Oleg Simakoff. 2015. A Comparative Analysis of IoT Platforms for the Medical Devices Industry. Dostopno na: http://www.ayantek.com/a-comparative-analysis-of-iot-platforms-for-the-medical-devices-industry/

[32] Miyuru Dayarathna. 2016. Comparing 11 IoT Development Platforms. Dostopno na: https://dzone.com/articles/iot-software-platform-comparison

[33] M&S Consulting. 2016. Industrial Internet of Things Platform Comparison. Dostopno na: http://www.mandsconsulting.com/industrial-iot-platform-comparison

[34] Standard Allseen Aliance. Dostopno na https://allseenalliance.org/

[35] OIC - Open Interconnect Consortium. Dostopno na: http://openinterconnect.org

[36] OneM2M - Standard za M2M in IoT komunikacijo. Dostopno na: http://www.onem2m.org/

[37] SmartM2M - Standard za M2M in IoT komunikacijo. Dostopno na: http://www.etsi.org/images/files/Events/2014/201405_DGCONNECT_SmartM2MAppliances/ETSI_M2M_introduction_main.pdf

[38] OBIX - Open Building Information Xchange. Dostopno na: http://www.obix.org/

[39] Torsten Grabs, Roman Schindlauer, Ramkumar Krishnan, Jonathan Goldstein. 2009. Introducing Microsoft StreamInsight. Dostopno na: http://web.cecs.pdx.edu/~tufte/410-510DS/readings_files/Microsoft%20CEP%20Overview.pdf

[40] Siddhi. Dostopno na: https://github.com/wso2/siddhi

[41] Licenca GNU General Public License 2.0. Dostopno na: https://opensource.org/licenses/gpl-2.0.php

[42] Licenca Apache 2.0. Dostopno na: http://www.apache.org/licenses/LICENSE-2.0.html

[43] Licenca EPL - Eclipse Public License 1.0. Dostopno na: https://opensource.org/licenses/eclipse-1.0.php

[44] Licenca MPL - Mozilla Public License 2.0. Dostopno na: https://www.mozilla.org/en-US/MPL/2.0/

[45] Eclipse za RCP in RAP razvijalce. Dostopno na: http://www.eclipse.org/downloads/packages/eclipse-rcp-and-rap-developers/heliosr

[46] Tycho konfigurator za orodje Eclipse. Dostopno na: `https://eclipse.org/tycho/`

[47] Primer vtičnika platforme OM2M, ki procesira vrednosti povezanega senzorja. Dostopno na: `https://wiki.eclipse.org/OM2M/one/Developer#Develop_an_Interworking_Proxy_Unit_.28IPE.29_plugin`

[48] GitHub repozitorij knjižnice om2m-cep.jar. Dostopno na: `https://github.com/gasperinn/om2m-cep/tree/master/library`

[49] Jurij Dobovišek, Rok Acceto. 2004. Arterijska hipertenzija. Dostopno na: `http://www.hipertenzija.org/pdf/Izjemno_nujna_in_nujna_stanja_arterijske_hipertenzije.pdf`

[50] GitHub. Dostopno na: `https://github.com`

[51] GitHub repozitorij vtičnika za procesiranje kompleksnih dogodkov. Dostopno na:

[52] Primer uporabe vtičnika s pomočjo grafičnega vmesnika. Dostopno na: `https://github.com/gasperinn/om2m-cep/blob/master/usage_example_gui.pdf`

[53] Primer uporabe knjižnice direktno v vtičniku platforme OM2M. Dostopno na: `https://github.com/gasperinn/om2m-cep/blob/master/usage_example_programatical_solution.pdf`

[54] Pogosta vprašanja o licenci EPL. Dostopno na: `https://eclipse.org/legal/eplfaq.php#GPLCOMPATIBLE`